**VŠB – Technická univerzita Ostrava**
**Fakulta elektrotechniky a informatiky**
**Katedra informatiky**

# J2ME tabulkový procesor – SmartSheet 2.0
# J2ME Spreadsheet – SmartSheet 2.0

2010 Pavel Kutáč

# Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. května 2010 ….....

Pavel Kutáč

# Abstrakt

Účelem této diplomové práce je rozšíření existující mobilní aplikace tabulkového procesoru SmartSheet o další funkcionalitu a úpravu programové a datové struktury pro efektivnější správu paměti. Nová funkcionalita je tvořena schopností importovat a exportovat soubory aplikace Microsoft Excel, včetně vzorců a formátování, a možností vytvářet uživatelské funkce pomocí vizuálního programování. Další změnou je úprava vlastního formátu souborů tak, aby byl pružnější a snáze se udržovala jeho kompatibilita se staršími verzemi tohoto souborového formátu. Poslední úpravy se týkají uživatelského rozhraní.



# Klíčová slova

tabulkový procesor, J2ME



# Abstract

The purpose of this diploma thesis is to extend existing mobile spreadsheet application SmartSheet with additional functionality and to change it's memory management to become more effective. New functionality consists of ability to import and export Microsoft Excel files including formulas and formatting, and possibility to create user functions using visual programming. Next goal is a change of native file format in such a way that it becomes more flexible and is easier to manage it's compatibility with older versions of this file format. Last changes are focused on user interface.



# Keywords

spreadsheet, J2ME

# Seznam použitých symbolů a zkratek

| | |
|---|---|
| **BIFF** | Binary Interchange File Format |
| **J2ME** | Java 2 Platform, Micro Edition |
| **MCD** | Microsoft Compound Document |
| **MSAT** | Master Sector Allocation Table |
| **RPN** | Reverse Polish Notation |
| **SAT** | Sector Allocation Table |
| **SSAT** | Short Sector Allocation Table |
| **XLS** | Přípona souborů aplikace Microsoft Excel |
| **XML** | Extensible Markup Language |

# Obsah

# 1 Úvod

Aplikace SmartSheet, vytvořená v rámci bakalářské práce nazvané „J2ME tabulkový procesor“, která představuje pokročilý tabulkový procesor pro mobilní zařízení na platformě J2ME, se ve své první verzi nevyhnula určitým nedostatkům, jenž sice nečinily tuto aplikaci nepoužitelnou, avšak její využití částečně omezovaly. Cílem této práce je tedy tyto nedostatky odstranit a přidat další funkcionalitu.

První změny se týkají programových a datových struktur. Správa paměti nebyla v předchozí verzi zcela optimální a zbývala spousta možností pro vylepšení. Z těchto možností byla pro úpravy zvolena změna způsobu reprezentace obsahu buněk, změna správy formátů buněk a správy textových hodnot buněk.

Dalším nedostatkem předchozí verze aplikace byla omezená možnost importu a exportu datových souborů pro přenos mezi obdobnými aplikacemi, která sestávala pouze ze schopnosti číst a zapisovat soubory CSV. Z tohoto důvodu je do aplikace přidána možnost importu a exportu binárních souborů aplikace Microsoft Excel, tedy souborů XLS. Kromě importu a exportu obsahu buněk, včetně vzorců, je podporován import a export formátování, což u souborů CSV nebylo možné.

Jako možnost rozšíření funkcionality aplikace uživatelem je přidána podpora vytváření uživatelských funkcí za pomoci jednoduchého vizuálního programování. Lze takto vytvářet funkce obsahující nejen standardní součásti vzorců, jakými jsou operátory a konstanty, ale také pokročilejší struktury, jako například podmínky a cykly. Možné je také využití proměnných při výpočtu výsledku funkce. K vytváření uživatelských funkcí slouží editor, kde má tvorba takové funkce z větší části zcela vizuální charakter, bez nutnosti se zabývat syntaxí nějakého textového programovacího jazyka.

Další nevýhodou předchozí verze aplikace byl její vlastní formát souborů, který byl příliš jednoduchý a zcela nerozšiřitelný při zachování určité míry kompatibility s jeho případnou původní verzí. Proto je vytvořen zcela nový formát, který je mnohem pružnější a jednodušší na rozšiřování, a který při dodržení několika zásad umožňuje jeho novým verzím zachovávat kompatibilitu se staršími verzemi aplikace.

Poslední změny se týkají uživatelského rozhraní, kde největším problémem předchozí verze aplikace bylo ovládání nevyužívající kontextových kláves a někdy zbytečně komplikovaný systém nabídek. Proto je systém nabídek přepracován, je přidána podpora kontextových kláves, a jsou vytvořeny nové komponenty pro vstupní formuláře, které slouží k pohodlnější interakci s uživatelem.

# 2 Změny v programové a datové struktuře

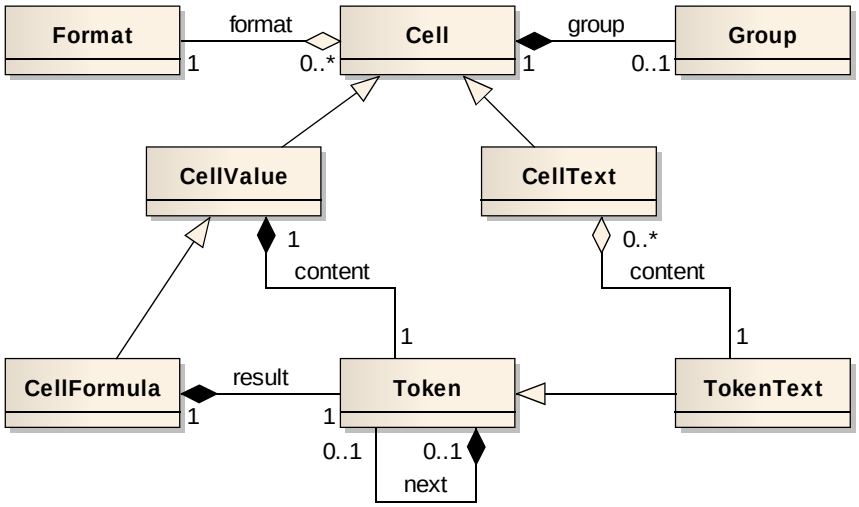
Cílem této kapitoly je popsat změny ve způsobu uchovávání dat v paměti a ve způsobu práce s těmito daty. K dispozici je také porovnání s předchozí verzí aplikace.

## 2.1 Důvody úprav datových struktur

Pro úpravu způsobu ukládání dat v paměti existovaly dva hlavní důvody, a to snížení paměťových nároků na uložení obsahu buněk a rozšíření funkcionality aplikace. Rozšíření aplikace o podporu importu/exportu dat z/do souborů XLS poskytovalo základní vzor pro návrh nového uspořádání datových struktur, neboť uspořádání dat v souborech XLS bylo zcela odlišné od původního uspořádání dat v paměti a pro usnadnění importu/exportu bylo výhodné se uspořádání dat v souborech XLS alespoň částečně přizpůsobit. Porovnání s předchozí verzí je v oddílu 2.7.

## 2.2 Reprezentace obsahu buněk

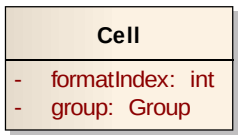
Pro reprezentaci různého obsahu buněk byla vytvořena hierarchie tříd, sestávající z tříd **Cell**, **CellText**, **CellValue** a **CellFormula** (viz obrázek 2). Třída **Format**, reprezentující formát buňky, byla upravena.




*Obrázek 1:* Diagram tříd reprezentujících obsah buněk

### 2.2.1 Třída Cell

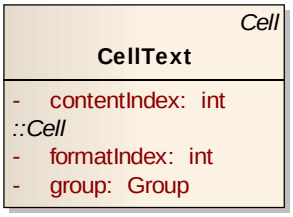
Třída reprezentuje buňku neobsahující žádnou hodnotu. Uchovává se pouze informace o formátování a sloučení buňky. Struktura třídy je popsána na obrázku 2. Atribut **formatIndex** obsahuje hodnotu indexu do tabulky formátů a atribut **group** odkazuje na objekt instance třídy **Group**, nesoucí informace o skupině sloučených buněk.




*Obrázek 2:* Třída Cell

### 2.2.2 Třída CellText

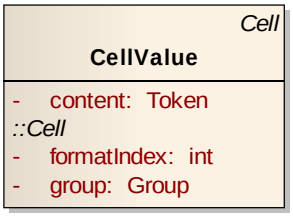
Tato třída, dědící z třídy **Cell**, reprezentuje buňku obsahující textovou hodnotu. Struktura třídy je popsána na obrázku 3. Atribut **contentIndex** obsahuje hodnotu indexu do tabulky textových řetězců.




*Obrázek 3:* Třída CellText

### 2.2.3 Třída CellValue

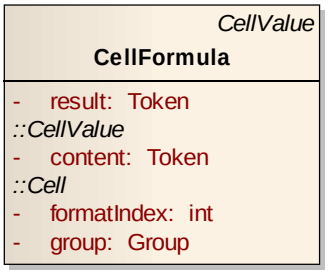
Tato třída, dědící z třídy **Cell**, reprezentuje buňku obsahující hodnotu jinou než textovou, přičemž druh hodnoty je určen objektem instance třídy dědící ze třídy **Token**. Struktura třídy je popsána na obrázku 4. Atribut **content** obsahuje odkaz na objekt instance třídy **Token** nesoucí hodnotu buňky.




*Obrázek 4:* Třída CellValue

### 2.2.4 Třída CellFormula

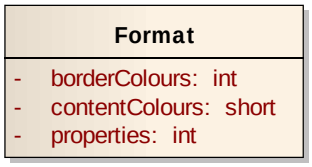
Tato třída, dědící z třídy **CellValue**, reprezentuje buňku obsahující vzorec a jeho výsledek. Struktura třídy je popsána na obrázku 5. Atribut **result** obsahuje odkaz na objekt instance třídy **Token** nesoucí hodnotu výsledku vzorce a atribut **content**, převzatý z rodičovské třídy **CellValue**, odkazuje na vzorec, jenž je tvořen zřetězeným seznamem objektů instance třídy **Token**.




*Obrázek 5:* Třída CellFormula

### 2.2.5 Třída Format

Tato třída, reprezentující formát buňky, byla přítomna již v předchozí verzi aplikace, ale s odlišným způsobem uložení hodnot formátování. Struktura třídy je popsána na obrázku 6. Atribut **properties** obsahuje nastavení formátování buňky, atribut **contentColours** obsahuje indexy do palety barev pro písmo a pozadí buňky a atribut **borderColours** obsahuje indexy do palety barev pro jednotlivé okraje buňky.




*Obrázek 6:* Třída Format

Jednotlivé informace o formátu buňky mají v rámci příslušných atributů přiděleny určité bity, do kterých jsou zakódovány. Způsob zakódování informací v rámci atributu **properties** je popsán v tabulce 1.

*Tabulka 1:* Přiřazení bitů v rámci atributu **properties**

| Bity | Obsah |
|---|---|
| 3-0 | Formát zobrazení čísla |
| Pokud je formát zobrazení čísla jiný než datum a čas | |
| 7-4 | Počet desetinných míst |
| 8 | Zvýraznění záporných hodnot |
| 9 | Zobrazení oddělovače tisíců |
| Pokud je formát zobrazení čísla datum a čas | |
| 5-4 | Formát zobrazení data |
| 6 | Zobrazení počátečních nulových číslic data |
| 7 | Zobrazení roku pomocí dvou číslic |
| 11-10 | Formát zobrazení času |
| 12 | Zobrazení počátečních nulových číslic času |
| 13 | Zobrazení 12-ti hodinového formátu času |
| 17-16 | Velikost písma |
| 21-18 | Styl písma |
| 24-22 | Horizontální zarovnání |
| 27-25 | Vertikální zarovnání |
| 31-28 | Zobrazení okrajů |

Význam bitů 15-4 atributu **properties** je závislý na hodnotě bitů 3-0 téhož atributu, přičemž některé zůstávají nevyužity. V rámci formátu buňky nejsou informace o barvách ukládány přímo jako hodnota barvy, ale jako index do palety barev. Má-li index zvláštní hodnotu $FF_H$, znamená to, že se jako index do palety barev použije ten, který je v aplikaci přednastavený jako index základní

barvy daného prvku formátu. Způsob zakódování informací v rámci atributu **contentColours** je popsán v tabulce 2.

*Tabulka 2:* Přiřazení bitů v rámci atributu **contentColours**

| Bity | Obsah |
|---|---|
| 7-0 | Index barvy písma |
| 15-8 | Index barvy pozadí |

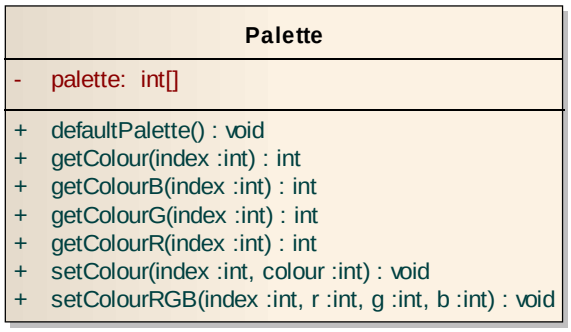
Způsob zakódování informací v rámci atributu **borderColours** je popsán v tabulce 3.

*Tabulka 3:* Přiřazení bitů v rámci atributu **borderColours**

| Bity | Obsah |
|---|---|
| 7-0 | Index barvy levého okraje |
| 15-8 | Index barvy horního okraje |
| 23-16 | Index barvy pravého okraje |
| 31-24 | Index barvy dolního okraje |

## 2.3 Paleta barev

Barvy písma, pozadí a okrajů se ve formátu buňky neuchovávají přímo jako hodnota, ale jako index do palety barev, která je reprezentována třídou **Palette** (viz obrázek 7). Paleta je tvořena 64 záznamy, přičemž každý záznam uchovává 24 bitovou RGB hodnotu (viz tabulka 4).




*Obrázek 7:* Třída Palette

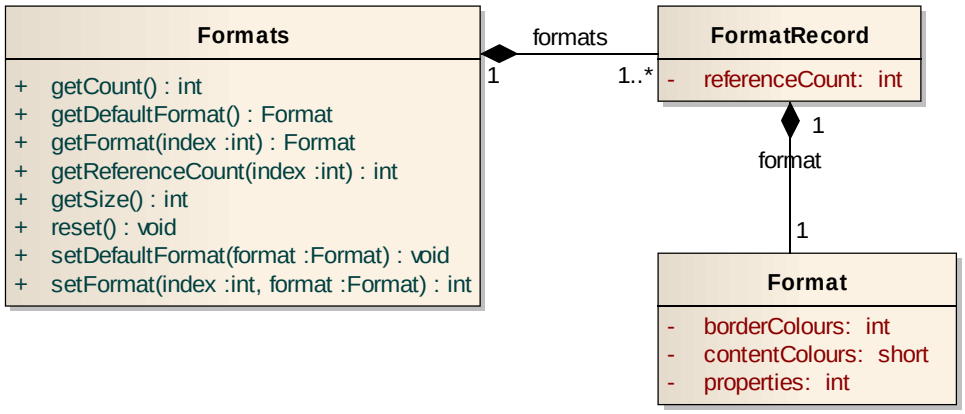
Metoda **defaultPalette** nastavuje paletu do základní podoby, poté co byla změněna pomocí **setColour** nebo **setColourRGB**. Metoda **getColour** vrací hodnotu barvy a metody **getColourR**, **getColourG** a **getColourB** vrací barevnou složku.

*Tabulka 4:* Přiřazení bitů v rámci záznamu palety barev

| Bity | Obsah |
|---|---|
| 7-0 | Modrá složka barvy |
| 15-8 | Zelená složka barvy |
| 23-16 | Červená složka barvy |

## 2.4 Správa formátů buněk

Buňky neobsahují přímo nastavení jejich formátu, ale pouze index do tabulky formátů, která je reprezentována třídou **Formats** (viz obrázek 8). Tato třída obsahuje vektor záznamů tvořených instancemi třídy **FormatRecord**, přičemž každý záznam vlastní jedinečnou instanci třídy **Format** (2.2.5), která uchovává samotné informace o formátování.




*Obrázek 8:* Diagram tříd reprezentujících správu formátů buněk

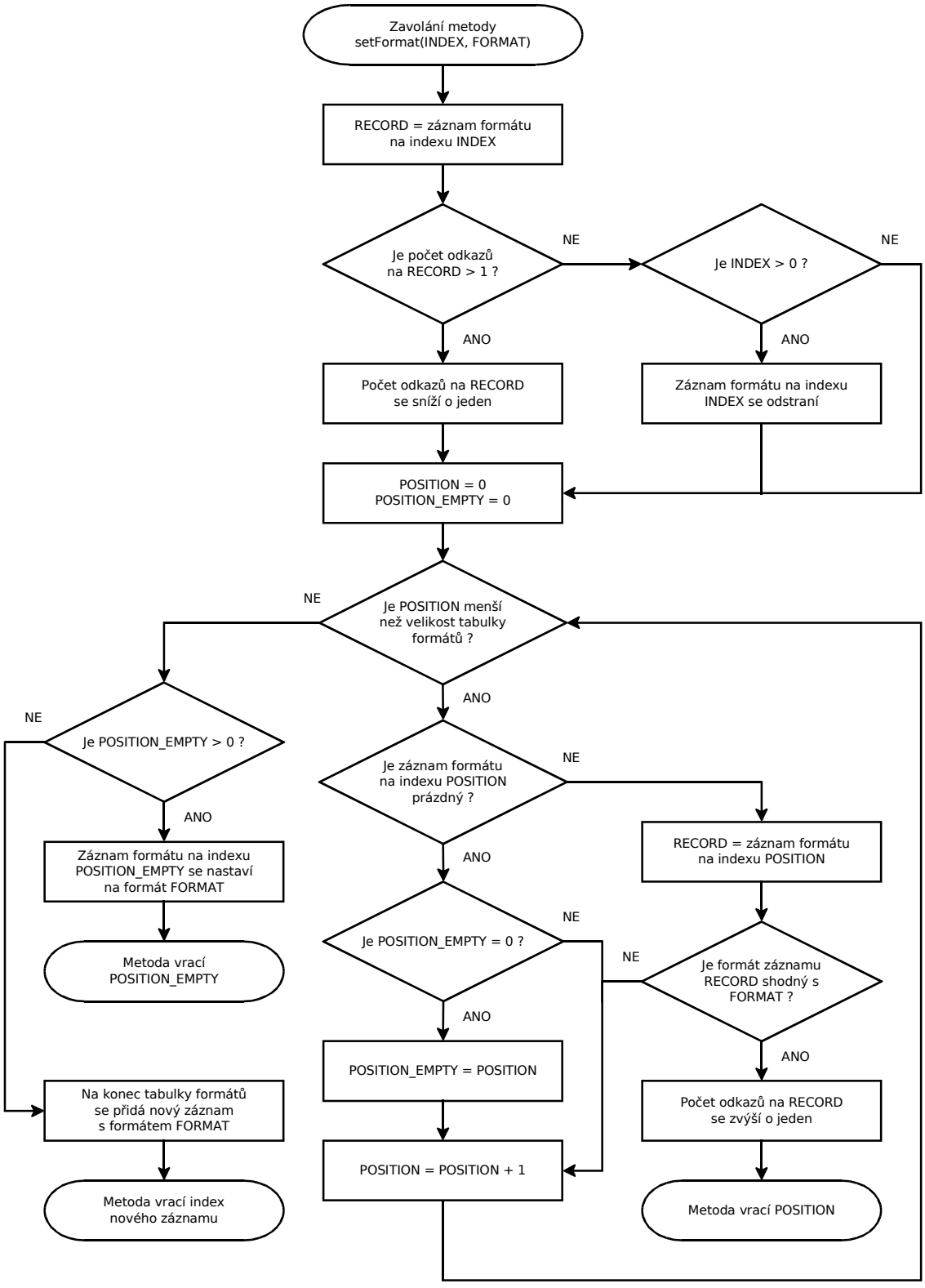
Třída **FormatRecord** dále obsahuje atribut **referenceCount**, který zaznamenává počet odkazů na obsažený formát, tedy kolik buněk dané formátování používá. Formáty buněk jsou tímto způsobem mezi buňkami sdílené a nemohou existovat dva záznamy v tabulce se stejným formátem.

V tabulce formátů zaujímá zvláštní místo záznam s indexem 0. Tento záznam nese základní formát, který je použit u buněk po jejich vytvoření, a v tabulce je vždy přítomen. Pro změnu základního formátu slouží metoda **setDefaultFormat** a pro jeho získání metoda **getDefaultFormat**.

Pro získání indexu formátu buňky slouží metoda **setFormat**. Tato metoda přijímá jako své parametry index předchozího formátu buňky a nový formát buňky. Její návratovou hodnotou je index nového formátu buňky. Index předchozího formátu slouží k určení záznamu v tabulce formátů, u něhož se provede snížení počtu odkazů. Snížení počtu odkazů má dva následky. V prvním případě je předchozí počet odkazů větší než 1 a záznam tak po snížení počtu odkazů zůstává v tabulce, jelikož na něj odkazují další buňky. V druhém případě je předchozí počet odkazů roven 1 a záznam je po snížení počtu odkazů z tabulky odstraněn, protože jej žádná další buňka nepoužívá, to ale pouze v případě formátu jiného než základního, ten v tabulce zůstává i s nulovým počtem odkazů. Při odstranění záznamu se následující záznamy v tabulce neposouvají, odstraněný záznam je pouze nahrazen prázdným ukazatelem.

Návratová hodnota metody **setFormat** je závislá na novém formátu buňky a na obsahu tabulky formátů. Formát, který se má pro buňku nastavit, již může být v tabulce přítomen, a proto je nutné tabulku nejprve prohledat. Tabulka se prohledává celá včetně základního formátu. Během prohledávání se také zjišťuje index prvního volného místa v tabulce, které mohlo zůstat po odstraněném záznamu. Pokud je nalezen záznam s formátem shodným s novým formátem, tak se prohledávání ukončí, nalezenému záznamu se zvýší počet odkazů o 1 a metoda vrátí jeho index jako index nového formátu buňky. Pokud není v tabulce nalezen záznam se shodným formátem, je do tabulky vložen nový záznam s novým formátem a počtem odkazů rovným 1, a to buď na první

volnou pozici, pokud byla nalezena, a nebo se přidá na konec tabulky. V obou případech metoda vrací index nového záznamu. Algoritmus metody **setFormat** je popsán na obrázku 9. Z předchozího textu plyne, že v tabulce formátů mohou zůstat prázdná místa mezi záznamy. Tato místa jsou odstraněna až při ukládání do souboru a indexy formátů jsou patřičně upraveny.



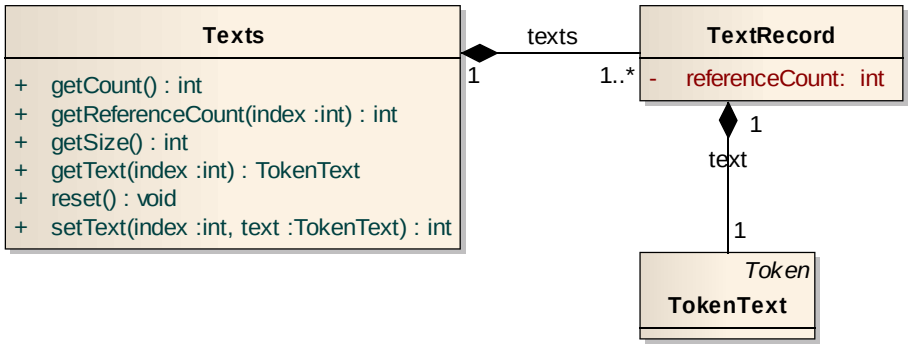

*Obrázek 9:* Algoritmus nastavení formátu buňky

Pro získání formátu s patřičným indexem slouží metoda **getFormat** a pro získání počtu odkazů metoda **getReferenceCount**.

Pro uvedení tabulky do výchozího stavu, kdy obsahuje pouze záznam se základním formátem, slouží metoda **reset**. Velikost tabulky se zjistí pomocí metody **getSize**, ale v tomto případě se do počtu záznamů započítávají také prázdná místa mezi záznamy. Skutečný počet záznamů, a tedy také formátů, vrací metoda **getCount**.

## 2.5 Správa textových hodnot

Buňky s textovou hodnotou neobsahují přímo danou hodnotu, ale pouze index do tabulky textových hodnot, která je reprezentována třídou **Texts** (viz obrázek 10). Tato třída obsahuje vektor záznamů tvořených instancemi třídy **TextRecord**, přičemž každý záznam vlastní jedinečnou instanci třídy **TokenText**, ve které je uložen samotný textový řetězec. Správa textových hodnot je používána ve spojení se třídou **CellText** (2.2.2).




*Obrázek 10:* Diagram tříd reprezentujících správu textových hodnot

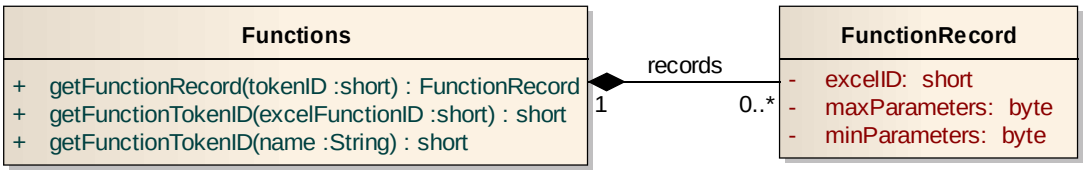
Správa textů je téměř shodná se správou formátů. Třída **TextRecord** také obsahuje atribut **referenceCount**, který zaznamenává počet odkazů na příslušný textový řetězec. Záznam s indexem 0 obsahuje prázdný řetězec a je v tabulce textových hodnot vždy přítomen.

Metoda **setText** slouží pro získání indexu textové hodnoty buňky. Tato metoda přijímá jako své parametry index předchozí textové hodnoty buňky a novou textovou hodnotu buňky, přičemž jediný rozdíl oproti metodě **setFormat** třídy **Formats** ve správě formátů (2.4) je ten, že se zde hledají a porovnávají textové řetězce namísto formátů, algoritmus je jinak zcela shodný. Odstranění mezer mezi záznamy a úprava indexů textových hodnot jsou také řešeny při ukládání do souboru. Metoda **getText** slouží pro získání textové hodnoty se zadaným indexem a metoda **getReferenceCount** vrací počet odkazů pro zadaný index textové hodnoty. Metody **reset**, **getSize** a **getCount** plní stejnou funkci jako ve správě formátů (2.4).

## 2.6 Tabulka funkcí

Důvodem pro vytvoření tabulky funkcí bylo přidání nové funkcionality do aplikace v podobě importu a exportu souborů XLS a vytváření uživatelských funkcí, kdy je potřeba mít k dispozici informace o identifikátorech či počtu parametrů.

Tabulka funkcí je reprezentována třídou **Functions** (viz obrázek 11), která obsahuje pro každou funkci záznam tvořený instancí třídy **FunctionRecord**.




*Obrázek 11:* Diagram tříd reprezentujících tabulku funkcí

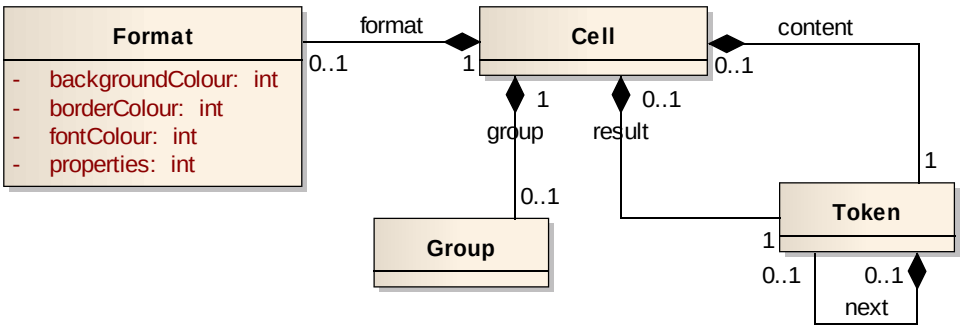
Třída **FunctionRecord** obsahuje atribut **excelID**, který uchovává identifikátor dané funkce v rámci aplikace Microsoft Excel, a dále atributy **minParameters** a **maxParameters**, které udávají nejmenší a největší počet parametrů pro danou funkci.

Třída **Functions** má metodu **getFunctionRecord**, vracející záznam funkce pro zadaný identifikátor funkce, a metodu **getFunctionTokenID**, která vrací identifikátor funkce pro zadaný identifikátor funkce aplikace Microsoft Excel nebo pro zadaný název funkce.

## 2.7 Porovnání s předchozí verzí aplikace

V předchozí verzi aplikace byla pro reprezentaci obsahu buňky přítomna pouze jediná třída, a to třída **Cell**. Tato třída obsahovala atribut **content** odkazující na objekt instance třídy **Token** nesoucí hodnotu buňky, nebo prázdný ukazatel v případě buňky bez hodnoty. Atribut **group** odkazoval na objekt instance třídy **Group**, popisující skupinu sloučených buněk. Dále třída vlastnila atribut **result** nesoucí odkaz na objekt instance třídy **Token** představující výsledek vzorce, nebo prázdný ukazatel v případě buňky bez vzorce.

Sdílení formátů a textových hodnot v předchozí verzi nebylo přítomno. Atribut **format** obsahoval odkaz na objekt instance třídy **Format**, popisující formátování buňky, nebo prázdný ukazatel v případě buňky se základním formátováním. Instance třídy **Format** byla pro každou buňku jedinečná, a to i v případě kdy měly buňky naprosto stejný formát. Stejně tak pokud dvě buňky obsahovaly shodný textový řetězec, jednalo se o dvě různé instance třídy **TokenText**.




*Obrázek 12:* Diagram tříd reprezentujících obsah buněk v předchozí verzi aplikace

Třída **Format** měla atribut **properties**, který se od stejnojmenného atributu v současné verzi třídy **Format** (2.2.5) lišil tím, že obsahoval v jeden okamžik informace o formátu zobrazení jak číselných hodnot tak i hodnot data a času, čímž se zmenšil prostor pro uchování dalších informací. Dále obsahoval atributy **fontColour**, **backgroundColour** a **borderColour**, poskytující hodnotu barev pro písmo, pozadí a okraj buňky. Ve formátu buňky byla uložena vždy přímo hodnota barvy, protože neexistovala paleta barev. Barva okraje byla pro všechny čtyři okraje buňky společná.

# 3 Import a export souborů XLS

Tato kapitola se zabývá importem a exportem souborů XLS. Je zde popsána struktura těchto souborů, použité třídy a algoritmy.

## 3.1 Charakteristika a historie souborů XLS

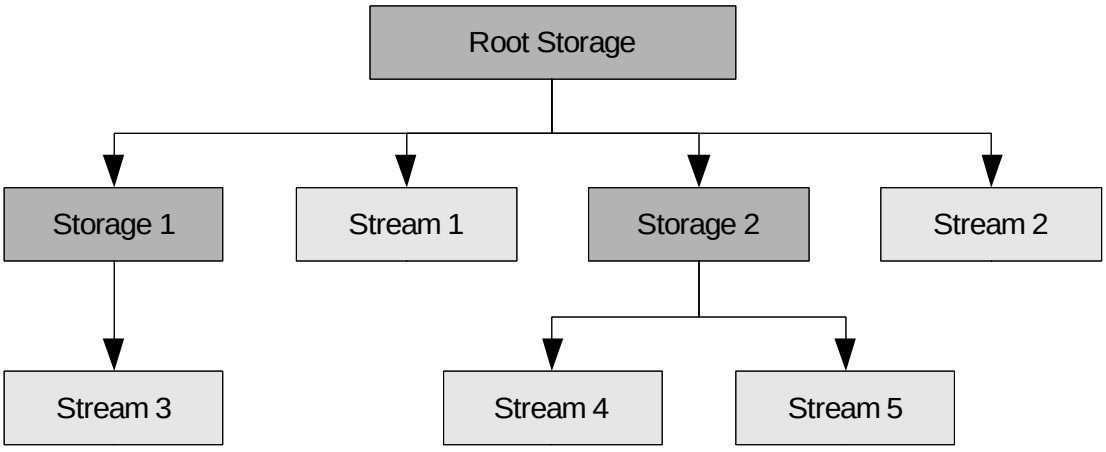
Soubor XLS je typ binárního souboru používaného aplikací Microsoft Excel. Byl primárním typem souborů pro ukládání dat až do verze Microsoft Excel 2007, kdy byl jako primární uveden nový typ souborů založený na XML.

Ačkoliv je zde označení XLS uváděno jako typ souboru, jedná se pouze o příponu, která označuje soubory tabulkového procesoru Microsoft Excel založené na binárním formátu zvaném BIFF (Binary Interchange File Format), který má několik verzí a jeho historie započala již v roce 1987. Verze BIFF2 až BIFF4 uchovávaly pouze jeden list, zatímco verze BIFF5 až BIFF8, což je také poslední verze, umožňují uchovávat více listů. Tento text se zabývá pouze poslední verzí formátu, označenou BIFF8. Tato verze je také jako jediná podporována aplikací SmartSheet v rámci importu a exportu. Vlastností této verze je skutečnost, že samotný soubor není tvořen pouze daty v BIFF formátu, ale tato data jsou zapouzdřena v rámci Microsoft Compound Document (MCD) souborovém formátu. Stručný popis struktury souboru XLS je na obrázku 14.

Tento typ souborů aplikace Microsoft Excel byl zvolen pro import a export z důvodu nižších nároků na systémové prostředky při zpracování oproti novější variantě založené na XML a také s ohledem na podporu starších verzí aplikace Microsoft Excel.

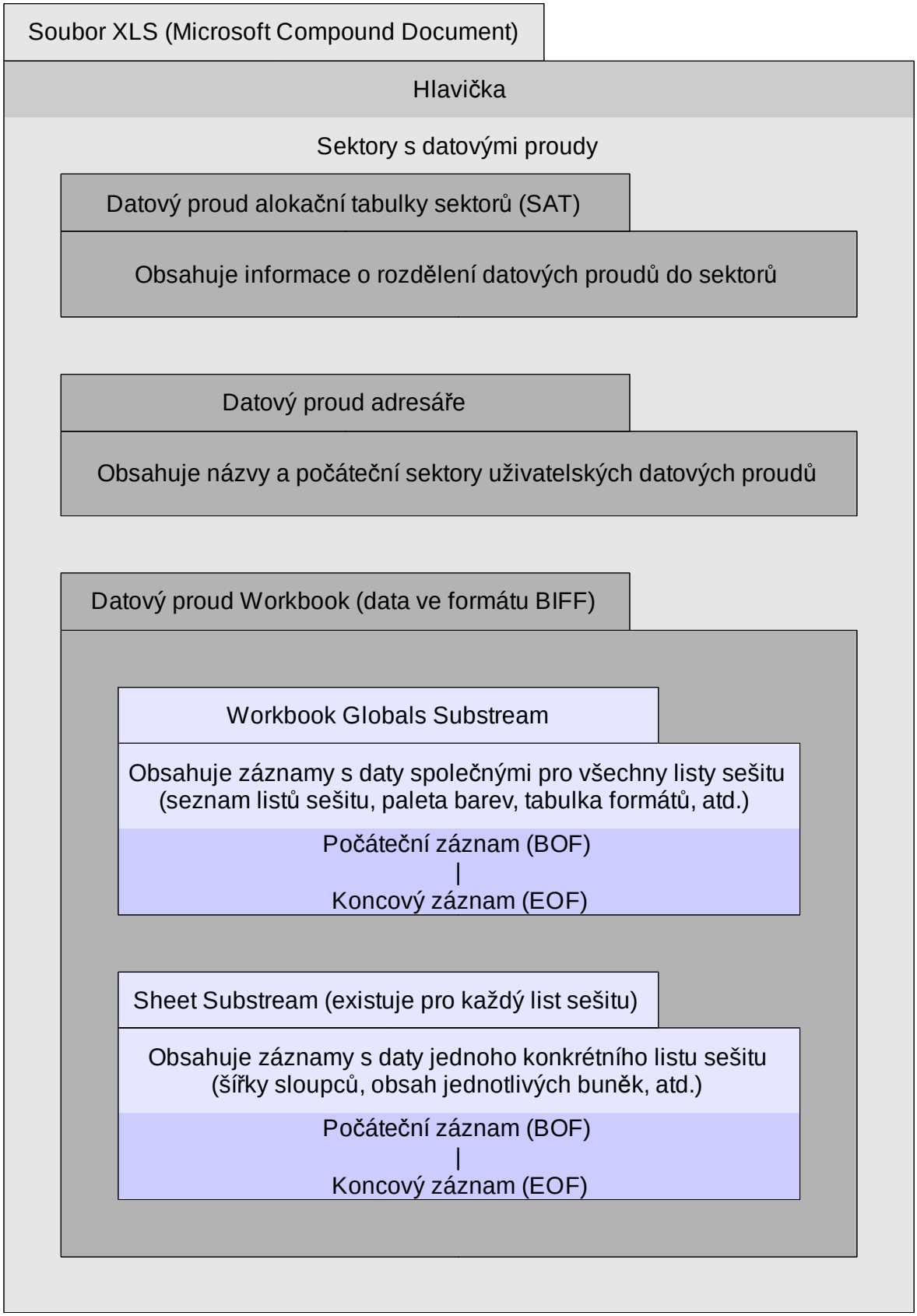
## 3.2 Microsoft Compound Document

Souborový formát umožňující v rámci jediného souboru uchovávat více datových celků. Toho je dosaženo strukturou souboru podobající se souborovému systému, sestávající z nezávislých datových proudů (stream), které jsou dále organizovány v hierarchii úschoven (storage).




*Obrázek 13:* Příklad hierarchie uvnitř Microsoft Compound Document

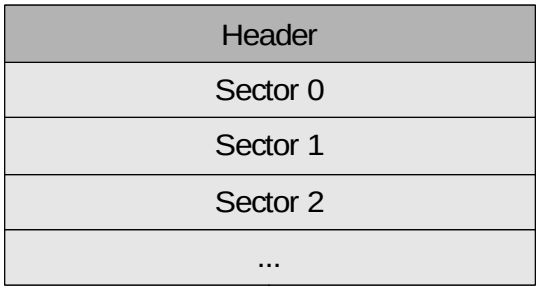
Datové proudy a úschovny mají názvy, které musí být odlišné pokud jsou dané objekty přímými členy stejné úschovny, ale objekty umístěné v různých úschovnách mohou mít názvy shodné. Na vrcholu hierarchie je vždy přítomna kořenová úschovna (root storage), viz obrázek 13.




*Obrázek 14:* Stručný popis struktury souboru XLS

### 3.2.1 Sektory a řetězce sektorů

Všechny datové proudy jsou rozděleny do menších bloků dat, nazvaných sektory (sector). Ty mohou obsahovat řídící data pro MCD nebo uživatelská data. Celý soubor MCD pak sestává z hlavičky (header) a po ní následujících sektorů (viz obrázek 15). Velikost sektoru je nastavena v hlavičce a platí pro všechny sektory v souboru.



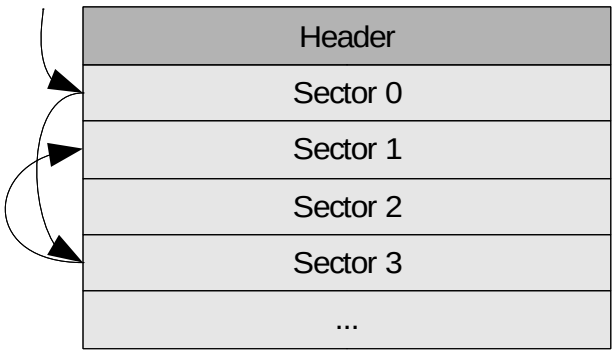

*Obrázek 15:* Rozdělení MCD na hlavičku a sektory

Sektory jsou očíslovány jejich pořadím v souboru, počínaje nulou. Tento index se nazývá identifikátor sektoru a je reprezentován 32bitovou celočíselnou hodnotou. Kladný identifikátor odkazuje na existující sektor, ale pokud je identifikátor sektoru záporný, pak má jeden ze zvláštních významů uvedených v tabulce 5.

*Tabulka 5:* Zvláštní významy identifikátoru sektoru

| Hodnota | Význam |
|---|---|
| -1 | Volný sektor, který není součástí žádného datového proudu |
| -2 | Ukončovací identifikátor v řetězci identifikátorů sektoru |
| -3 | Sektor je použit pro alokační tabulku sektorů |
| -4 | Sektor je použit pro hlavní alokační tabulku sektorů |

Seznam všech sektorů použitých pro uložení dat určitého datového proudu se nazývá řetězec sektorů (sector chain). Části datového proudu mohou být v souboru MCD uloženy neuspořádaně, takže pole identifikátorů sektoru, neboli řetězec sektorů, určuje pořadí sektorů v jakém má být datový proud přečten. Řetězec sektorů je vždy zakončen hodnotou -2.




*Obrázek 16:* Příklad řetězce sektorů

V příkladu na obrázku 16 je datový proud uložen ve třech sektorech počínaje sektorem 0 a jeho řetězec sektorů má podobu [0, 3, 1, -2].

### 3.2.2 Hlavička

Hlavička je vždy umístěna na začátku souboru MCD a její velikost je 512 byte. Obsahuje informace nutné pro čtení obsahu souboru MCD, včetně první části hlavní alokační tabulky sektorů. Struktura hlavičky je popsána v tabulce 6. Pomocí údajů z hlavičky lze vypočítat pozici (offset) sektoru v rámci souboru MCD. Postup výpočtu je následující:

$$pozice\ sektoru = 512 + identifikátor\ sektoru \cdot velikost\ sektoru$$

*Tabulka 6:* Struktura hlavičky

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 0 | 8 | Identifikátor souboru MCD: $D0_H\ CF_H\ 11_H\ E0_H\ A1_H\ B1_H\ 1A_H\ E1_H$ |
| 8 | 16 | Jedinečný identifikátor souboru (UID) |
| 24 | 2 | Číslo revize formátu souboru |
| 26 | 2 | Číslo verze formátu souboru |
| 28 | 2 | Identifikátor pořadí byte:<br>$FE_H\ FF_H$ – little-endian<br>$FF_H\ FE_H$ – big-endian |
| 30 | 2 | Po umocnění čísla 2 touto hodnotou získáme velikost sektoru v jednotkách byte (nejmenší možná velikost hodnoty je 7, což znamená velikost sektoru 128 byte) |
| 32 | 2 | Po umocnění čísla 2 touto hodnotou získáme velikost krátkého sektoru v jednotkách byte (největší možná velikost je velikost sektoru) |
| 34 | 10 | Nepoužito |
| 44 | 4 | Celkový počet sektorů použitých pro alokační tabulku sektorů |
| 48 | 4 | Identifikátor prvního sektoru datového proudu obsahujícího adresář |
| 52 | 4 | Nepoužito |
| 56 | 4 | Minimální velikost datového proudu v jednotkách byte (nejmenší možná hodnota je 4096 byte), kdy každý datový proud s velikostí menší než tato hodnota je uložen jako krátký datový proud |
| 60 | 4 | Identifikátor prvního sektoru alokační tabulky krátkých sektorů (hodnota -2 pokud tabulka není použita) |
| 64 | 4 | Celkový počet sektorů použitých pro alokační tabulku krátkých sektorů |
| 68 | 4 | Identifikátor prvního sektoru hlavní alokační tabulky sektorů (hodnota -2 pokud nejsou použity žádné dodatečné sektory) |
| 72 | 4 | Celkový počet sektorů použitých pro hlavní alokační tabulku sektorů |
| 76 | 436 | Část hlavní alokační tabulky sektorů obsahující prvních 109 identifikátorů sektorů |

### 3.2.3 Hlavní alokační tabulka sektorů

Hlavní alokační tabulka sektorů (MSAT) je pole identifikátorů všech sektorů použitých alokační tabulkou sektorů (SAT), která je potřebná pro čtení všech datových proudů uvnitř souboru MCD. SAT také tvoří datový proud a MSAT popisuje jak jej přečíst. Velikost MSAT, tedy počet identifikátorů sektoru, je shodný s počtem sektorů použitých pro SAT, přičemž tato hodnota je uložena v hlavičce.

Prvních 109 záznamů MSAT je uloženo v hlavičce. Jestliže MSAT obsahuje více než 109 záznamů, jsou následující záznamy uloženy v dodatečných sektorech. Hlavička v takovémto případě obsahuje identifikátor prvního dodatečného sektoru, jinak obsahuje hodnotu -2. Poslední identifikátor sektoru v každém dodatečném sektoru MSAT odkazuje na následující dodatečný sektor MSAT. Pokud žádný další sektor nenásleduje, má poslední záznam hodnotu -2. Poslední sektor MSAT nemusí být zcela využit. Nepoužité záznamy jsou poté naplněny hodnotou -1.

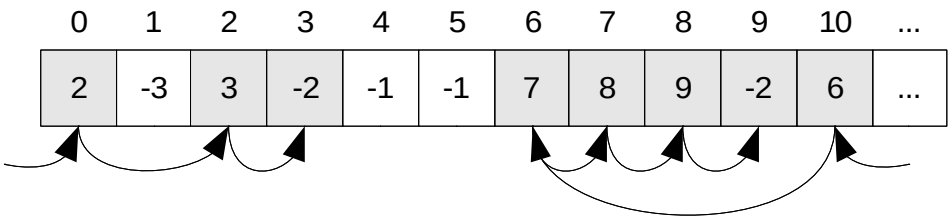
### 3.2.4 Alokační tabulka sektorů

Alokační tabulka sektorů (SAT) je pole identifikátorů sektoru. Obsahuje řetězce sektorů všech uživatelských datových proudů kromě krátkých a některých zvláštních datových proudů, jakými jsou kontejner pro krátké datové proudy, alokační tabulka krátkých sektorů a adresář. Velikost SAT, tedy počet identifikátorů sektoru, je shodný s celkovým počtem sektorů v souboru MCD.

Tabulka se sestaví tak, že se přečte a spojí obsah všech sektorů uvedených v MSAT. Sektory musí být čteny ve stejném pořadí v jakém jsou zapsány jejich identifikátory v MSAT.

Při čtení jednotlivých sektorů datových proudů pomocí SAT se postupuje tak, že aktuální index do SAT označuje aktuálně čtený sektor a hodnota na aktuálním indexu do SAT představuje identifikátor následujícího čteného sektoru a tudíž následující index do SAT. Alokační tabulka má tedy pro každý sektor v souboru MCD záznam s indexem shodným s identifikátorem daného sektoru a v tomto záznamu je uveden identifikátor následujícího sektoru neboli následující index do SAT. Pokud za aktuálním sektorem již žádný další nenásleduje, je v SAT na aktuálním indexu uvedena hodnota -2. Nepoužité sektory mají v SAT záznam o hodnotě -1. Sektory použité pro SAT nejsou nijak zřetězeny a mají hodnotu -3, stejně tak sektory použité pro MSAT nejsou zřetězeny a mají hodnotu -4.

Počáteční index do SAT pro určitý uživatelský datový proud se získává z adresáře, v případě řídících datových proudů naopak z hlavičky.




*Obrázek 17:* Příklad alokace sektorů

V příkladu na obrázku 17 jsou dva datové proudy. První z nich začíná v sektoru 0 a jeho řetězec sektorů je [0, 2, 3, -2], druhý začíná v sektoru 10 a má řetězec sektorů [10, 6, 7, 8, 9, -2]. Pro SAT je využit sektor 1, sektory 4 a 5 jsou nepoužité.

### 3.2.5 Krátké datové proudy

Kdykoliv je datový proud kratší nežli minimální velikost zapsaná v hlavičce, je uložen jako krátký datový proud. Krátké datové proudy nepoužívají pro uložení svého obsahu přímo sektory, ale jsou sdruženy do zvláštního datového proudu, nazvaného kontejner krátkých datových proudů.

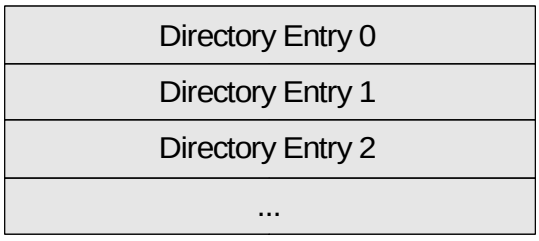
Kontejner krátkých datových proudů je uložen stejně jako další datové proudy. První použitý sektor se získá z kořenového záznamu v adresáři a řetězec sektorů se zjistí ze SAT. Rozdíl oproti ostatním datovým proudům je ten, že tento je dále rozdělen na krátké sektory (short sector), a to podobným způsobem jako celý soubor MCD na sektory. První krátký sektor má tedy identifikátor s hodnotou 0 a je umístěn vždy na začátku kontejneru. Po tomto následují další se vzestupnou hodnotou identifikátoru. Velikost krátkého sektoru je uvedena v hlavičce a díky tomu lze vypočítat pozici (offset) krátkého sektoru v rámci kontejneru krátkých sektorů následujícím způsobem:

$$pozice\ krátkého\ sektoru = identifikátor\ krátkého\ sektoru \cdot velikost\ krátkého\ sektoru$$

Alokační tabulka krátkých sektorů (SSAT) je pole identifikátorů krátkého sektoru a obsahuje řetězce krátkých sektorů pro všechny krátké datové proudy. Tabulka je uložena stejně jako ostatní řídící datové proudy. První použitý sektor je uveden v hlavičce a řetězec sektorů se získá ze SAT. Tabulka se sestaví tak, že se přečte a spojí obsah všech sektorů podle řetězce v SAT. Používá se stejně jako SAT s rozdílem, že obsahuje identifikátory krátkých sektorů odkazující do kontejneru pro krátké datové proudy.

### 3.2.6 Adresář

Adresář (directory) je řídící datový proud, který se skládá z pole záznamů adresáře (directory entry). Každý záznam přísluší určitému datovému proudu nebo úschovně v souboru MCD. Záznamy jsou číslovány v pořadí jejich výskytu v adresáři, počínaje hodnotou 0 (viz obrázek 18). Toto označení se nazývá identifikátor záznamu adresáře.




*Obrázek 18:* Rozdělení adresáře na záznamy

Záznam s identifikátorem o hodnotě 0, tedy první záznam v adresáři, vždy reprezentuje kořenovou úschovnu (root storage). Pokud je některý datový proud ze souboru MCD odstraněn, patřičný záznam v adresáři zůstává, pouze se označí jako prázdný.

Adresář organizuje položky (datové proudy a úschovny) jednotlivých úschoven do samostatných red-black stromů, přičemž položky jsou porovnávány podle svých jmen.

Záznam adresáře má vždy velikost 128 byte a pozici (offset) záznamu v rámci datového proudu adresáře lze vypočítat následujícím způsobem:

$$pozice\ záznamu\ adresáře = identifikátor\ záznamu\ adresáře \cdot 128$$

Struktura záznamu adresáře je popsána v tabulce 7. Při exportu se na místo časových značek vytvoření a poslední změny záznamu zapisují pouze nuly. Záznam adresáře pro datový proud obsahuje identifikátor počátečního sektoru, který slouží jako výchozí bod pro čtení obsahu datového proudu pomocí SAT. Pokud se jedná o krátký datový proud, pak záznam obsahuje identifikátor počátečního krátkého sektoru v rámci kontejneru krátkých datových proudů a ke čtení se použije SSAT.

*Tabulka 7:* Struktura záznamu adresáře

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** |
|---|---|---|
| 0 | 64 | Jméno záznamu ukončené nulovým znakem (použito 16bitové Unicode kódování) |
| 64 | 2 | Počet byte použitých pro jméno záznamu včetně nulového znaku (nejedná se o počet znaků, ale o dvojnásobek počtu znaků) |
| 66 | 1 | Typ záznamu:<br>$00_H$ – prázdný<br>$01_H$ – uživatelská úschovna<br>$02_H$ – uživatelský datový proud<br>$05_H$ – kořenová úschovna |
| 67 | 1 | Barva záznamu v rámci red-black stromu:<br>$00_H$ – červená<br>$01_H$ – černá |
| 68 | 4 | Identifikátor záznamu levého potomka v rámci red-black stromu všech potomků rodičovské úschovny, hodnota -1 pokud levý potomek chybí |
| 72 | 4 | Identifikátor záznamu pravého potomka v rámci red-black stromu všech potomků rodičovské úschovny, hodnota -1 pokud pravý potomek chybí |
| 76 | 4 | Identifikátor kořenového záznamu v rámci red-black stromu všech členů úschovny pokud záznam představuje úschovnu, jinak hodnota -1 |
| 80 | 16 | Jedinečný identifikátor (pokud záznam představuje úschovnu) |
| 96 | 4 | Nepoužito |
| 100 | 8 | Časová značka vytvoření záznamu |
| 108 | 8 | Časová značka poslední změny záznamu |
| 116 | 4 | Identifikátor prvního sektoru nebo prvního krátkého sektoru pokud záznam reprezentuje datový proud, identifikátor prvního sektoru kontejneru krátkých datových proudů pokud záznam reprezentuje kořenovou úschovnu, jinak hodnota 0 |
| 120 | 4 | Celková velikost datového proudu v jednotkách byte pokud záznam reprezentuje datový proud, celková velikost kontejneru krátkých datových proudů v jednotkách byte pokud záznam reprezentuje kořenovou úschovnu, jinak hodnota 0 |
| 124 | 4 | Nepoužito |

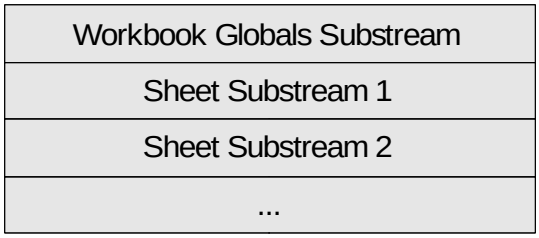
### 3.2.7 Vlastní řešení čtení a zápisu

Při importu je vždy celý aktuální sektor načten v paměti, kde se z něj sekvenčním přístupem získávají jednotlivé byte. Po přečtení hodnoty posledního byte aktuálního sektoru se načte namísto něj obsah následujícího sektoru. Kromě aktuálního sektoru je v paměti po celou dobu čtení dat uložena také celá SAT a SSAT. Adresář není potřeba uchovávat, jelikož se čte obsah jediného datového proudu, nazvaného Workbook. Stejně tak MSAT je potřeba pouze pro vytvoření SAT.

Při exportu jsou nejdříve zapsány nulové byte místo hlavičky a prvního sektoru určeného pro adresář. Důvodem je neznalost mnohých údajů na počátku exportu, jako například počáteční sektory alokačních tabulek sektorů. Poté se zapisují uživatelská data tvořící datový proud Workbook, přičemž tento datový proud je na konci případně doplněn nulovými byte, tak aby bylo možné jej rozdělit na stejně velké sektory. Poté je zapsána SSAT, pokud je použit krátký datový proud. Následuje zápis SAT a MSAT. Nakonec je zapsán skutečný obsah hlavičky a adresáře.

## 3.3 Binary Interchange File Format

Binární formát sloužící k uložení uživatelských dat aplikace Microsoft Excel. Má podobu datového proudu rozděleného na dílčí proudy dat. První dílčí proud obsahuje údaje společné pro celý soubor (Workbook Globals Substream). Po něm následují dílčí proudy dat jednotlivých listů sešitu (Sheet Substream), přičemž v souboru musí být přítomen alespoň jeden list (viz obrázek 19).




*Obrázek 19:* Příklad rozdělení na dílčí proudy dat

Tento text se věnuje výhradně poslední verzi tohoto formátu s označením BIFF8, což je také jediná verze, kterou lze v aplikaci SmartSheet importovat.

### 3.3.1 Záznamy

Uživatelská data v jednotlivých dílčích proudech jsou uložena v podobě záznamů se společnou základní strukturou (viz tabulka 8). Ta je tvořena hlavičkou, která obsahuje identifikátor záznamu a velikost dat záznamu v jednotkách byte, následovanou daty záznamu. Identifikátor určuje typ záznamu a druh obsažených dat. Do hodnoty velikosti záznamu se nezapočítává velikost hlavičky.

*Tabulka 8:* Základní struktura záznamu

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** |
|---|---|---|
| 0 | 2 | Identifikátor záznamu |
| 2 | 2 | Velikost dat záznamu (**sz**) |
| 4 | **sz** | Data záznamu |

Údaj o velikosti dat záznamu je při importu použit pro přeskočení neznámých záznamů. Největší možná velikost dat záznamu je 8224 byte, kdy po dosažení této hodnoty jsou další data záznamu uložena do jednoho či více následujících záznamů typu **CONTINUE**. Indexy sloupců a řádků, použité v některých záznamech, mají počátek v nule.

### 3.3.2 Řetězce znaků

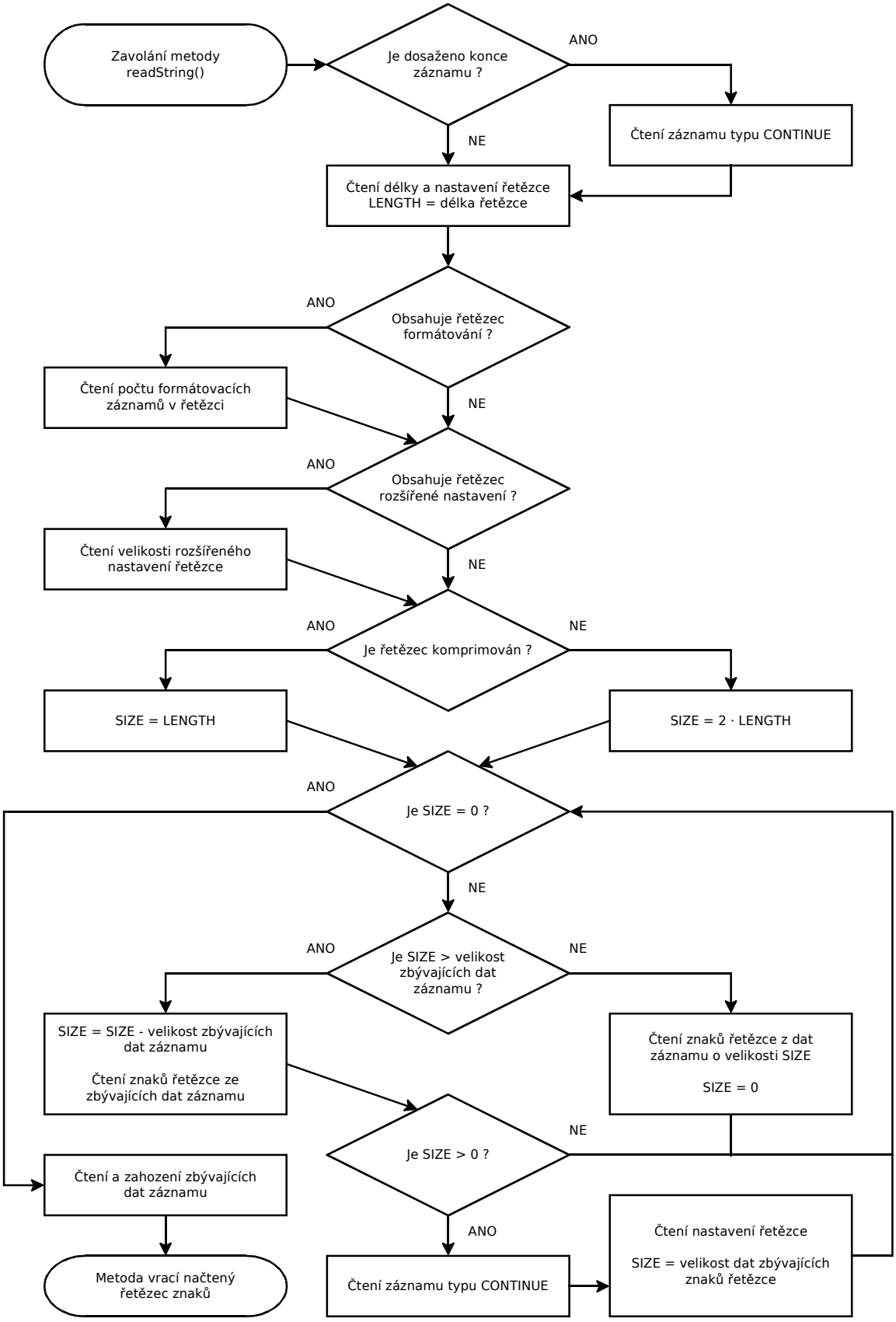
Ve verzi BIFF8 je pro ukládání znakových řetězců použito kódování UTF-16LE v rámci zvláštního formátu řetězců popsaného v tabulce 9.

*Tabulka 9:* Formát řetězce

<table>
<tr><th>Pozice (offset)</th><th>Velikost (byte)</th><th colspan="2">Obsah</th></tr>
<tr><td>0</td><td>1 nebo 2</td><td colspan="2">Počet znaků v řetězci (ln)</td></tr>
<tr><td rowspan="4">1 nebo 2</td><td rowspan="4">1</td><td>Bity</td><td>Obsah</td></tr>
<tr><td>0</td><td>Komprese znaků (cc):<br>$0_B$ – znaky jsou komprimovány<br>$1_B$ – znaky nejsou komprimovány</td></tr>
<tr><td>2</td><td>Rozšířené nastavení řetězce (ext):<br>$0_B$ – neobsahuje rozšířené nastavení řetězce<br>$1_B$ – obsahuje rozšířené nastavení řetězce</td></tr>
<tr><td>3</td><td>Formátování řetězce (rt):<br>$0_B$ – neobsahuje formátování řetězce<br>$1_B$ – obsahuje formátování řetězce</td></tr>
<tr><td>2 nebo 3</td><td>2</td><td colspan="2">[přítomno pouze pokud rt = $1_B$]<br>Počet formátovacích záznamů v řetězci (rtcnt)</td></tr>
<tr><td>proměnná</td><td>4</td><td colspan="2">[přítomno pouze pokud ext = $1_B$]<br>Velikost rozšířeného nastavení řetězce v jednotkách byte (extsz)</td></tr>
<tr><td>proměnná</td><td>ln<br>nebo 2·ln</td><td colspan="2">Pole znaků řetězce (v závislosti na nastavení cc jsou použity 8bitové nebo 16bitové znaky)</td></tr>
<tr><td>proměnná</td><td>4·rtcnt</td><td colspan="2">[přítomno pouze pokud rt = $1_B$]<br>Seznam formátovacích záznamů v řetězci</td></tr>
<tr><td>proměnná</td><td>extsz</td><td colspan="2">[přítomno pouze pokud ext = $1_B$]<br>Rozšířené nastavení řetězce</td></tr>
</table>

Počet znaků v řetězci může být uložen jako 8bitová nebo 16bitová hodnota. Která z variant se použije závisí na typu záznamu. Použité Unicode kódování znaků vyžaduje k uložení jednoho znaku 16 bitů. Jestliže všechny znaky v řetězci mají horní byte nulový a v nastavení řetězce je povolena komprese znaků, je u všech znaků horní byte opomíjen a ty jsou poté uloženy jako 8bitové hodnoty.

Jsou-li data řetězce rozdělena do dalšího záznamu typu **CONTINUE**, údaj o počtu znaků se již v dalším záznamu neuvádí, avšak údaj o nastavení řetězce ano. Tento údaj obsahuje pouze informaci o kompresi dané části řetězce, takže části řetězce v jednotlivých navazujících záznamech mohou být komprimovány nezávisle na sobě. Postup čtení řetězce znaků zachycuje obrázek 20.




*Obrázek 20:* Algoritmus čtení řetězce znaků

Při čtení řetězce v rámci importu se nejprve zjišťuje, zda nebylo dosaženo konce dat záznamu. V takovém případě řetězec začíná v navazujícím záznamu typu **CONTINUE**. Jako první je přečten údaj o délce řetězce, reprezentovaný 8bitovou nebo 16bitovou hodnotou v závislosti na typu záznamu, následovaný údajem o nastavení řetězce. Z tohoto údaje se zjistí zda je řetězec komprimovaný, zda obsahuje rozšířené nastavení či formátování. Informace o komprimaci řetězce umožňuje vypočítat velikost dat obsahujících znaky řetězce. Pokud je velikost dat řetězce větší než velikost dat záznamu zbývajících ke zpracování, jsou zbylá data záznamu zpracována a poté se očekává přítomnost navazujícího záznamu typu **CONTINUE**, který obsahuje další část řetězce. Tímto způsobem se pokračuje do doby, než jsou přečteny všechny znaky řetězce. Zbylá data v posledním záznamu jsou přeskočena.

Při zápisu řetězce v rámci exportu je brán ohled na skutečnost, že data řetězce mohou být rozdělena pouze takovým způsobem, aby první záznam obsahoval pohromadě všechny údaje hlavičky řetězce a alespoň jeden znak. Zapisovány jsou pouze velikost, nastavení a znaky řetězce. Před zápisem se nejprve zjišťuje možnost komprimace řetězce prohledáním všech jeho znaků, přičemž zjištěné nastavení je použito pro celý řetězec, včetně částí zapsaných v navazujících záznamech typu **CONTINUE**.

Rozšířené nastavení řetězce se týká pouze některých jazyků a při importu a exportu jsou tato nastavení přehlížena. Formátování řetězce označuje změnu vlastností písma pro části řetězce znaků pomocí formátovacích záznamů. Informace o formátování řetězce jsou při importu a exportu rovněž přehlíženy.

### 3.3.3 Záznamy pro organizaci dílčích proudů dat

Záznamem typu **BOF** (viz tabulka 10) začíná každý dílčí proud a obsahuje především informace o typu proudu.

*Tabulka 10:* Struktura záznamu typu **BOF**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | 2 | Verze formátu BIFF ($0600_H$ pro BIFF8) |
| 6 | 2 | Typ dílčího proudu dat:<br>$05_H$ – Workbook Globals Substream<br>$10_H$ – Sheet Substream |
| 8 | 2 | Identifikátor použité verze Microsoft Excel ($0DBB_H$ – Excel 97) |
| 10 | 2 | Rok vydání použité verze Microsoft Exdel ($07CC_H$ – Excel 97) |
| 12 | 4 | Historie zápisu souboru (používá pouze Microsoft Excel) |
| 16 | 4 | Nejnižší verze BIFF formátu platná pro všechny záznamy v souboru |

Při vlastním importu je většina informací přehlížena, kontroluje se pouze verze BIFF formátu a typ dílčího proudu dat, přičemž typy jiné než které jsou uvedené v tabulce 10, jsou při importu přehlíženy. Při exportu jsou do záznamu zapsány takové hodnoty, aby se daný soubor prezentoval jako soubor zapsaný prostřednictvím Microsoft Excel 97 ve formátu BIFF8.

Záznam typu **EOF** je na konci každého dílčího proudu dat a značí konec proudu. Tento záznam neobsahuje žádná data.

Záznam typu **BOUNDSHEET** (viz tabulka 11) existuje v rámci Workbook Globals Substream pro každý Sheet Substream v souboru. Každý takový záznam obsahuje absolutní pozici prvního záznamu daného listu sešitu od počátku celého datového proudu Workbook a další informace.

*Tabulka 11:* Struktura záznamu typu **BOUNDSHEET**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** |
|---|---|---|
| 4 | 4 | Absolutní pozice počátku záznamu typu BOF, uvozujícího dílčí proud daného listu sešitu, od počátku datového proudu Workbook |
| 8 | 1 | Stav listu sešitu:<br>$00_H$ – Viditelný<br>$01_H$ – Skrytý<br>$02_H$ – Programově skrytý |
| 9 | 1 | Typ listu sešitu ($00_H$ – Worksheet) |
| 10 | proměnná | Jméno listu sešitu (Unicode řetězec s 8bitovou délkou, viz 3.3.2) |

V aplikaci SmartSheet lze importovat listy sešitu viditelné i skryté, včetně programově skrytých. Posledního jmenovaného stavu listu lze dosáhnout pomocí procedury jazyka Visual Basic, a takto skrytý list je možné v aplikaci Microsoft Excel zviditelnit opět pouze přes proceduru jazyka Visual Basic, nikoliv pomocí nabídky uživatelského rozhraní. V aplikaci SmartSheet tato možnost není, a proto jsou všechny listy k dispozici jako viditelné. Ze všech typů listů sešitu lze importovat pouze Worksheet, jenž značí list obsahující tabulku.

### 3.3.4 Záznam reprezentující paletu barev

Záznam typu **PALETTE** (viz tabulka 12) obsahuje hodnoty uživatelské palety barev. Pokud tento záznam není ve Workbook Globals Substream přítomen, znamená to, že má být použita základní paleta barev.

*Tabulka 12:* Struktura záznamu typu **PALETTE**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** | |
|---|---|---|---|
| 4 | 2 | Počet následujících hodnot uživatelské palety barev (**cnt**) | |
| 6 | 4·**cnt** | **Hodnota barvy** | |
| | | **Bity** | **Obsah** |
| | | 7-0 | Červená složka barvy |
| | | 15-8 | Zelená složka barvy |
| | | 23-16 | Modrá složka barvy |

Jednotlivé barvy v paletě mají index s počátkem v nule, ale hodnoty barev obsažené v tomto záznamu se ukládají do palety od indexu 8, protože prvních osm barev palety je pevně daných.

### 3.3.5 Záznamy obsahující údaje o parametrech sešitu

Záznam typu **DIMENSIONS** (viz tabulka 13) obsahuje údaje o rozsahu využité části tabulky listu sešitu. Je obsažen v každém Sheet Substream.

*Tabulka 13:* Struktura záznamu typu **DIMENSIONS**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | 4 | Index prvního použitého řádku |
| 8 | 4 | Index posledního použitého řádku, zvýšený o jedna |
| 12 | 2 | Index prvního použitého sloupce |
| 14 | 2 | Index posledního použitého sloupce, zvýšený o jedna |
| 16 | 2 | Nepoužito |

Záznam typu **WINDOW1** (viz tabulka 14) je obsažen ve Workbook Globals Substream a jsou v něm zapsány vlastnosti okna dokumentu. Při exportu je tento záznam zapsán s takovými údaji, aby okno dokumentu bylo po otevření souboru viditelné, včetně obou posuvníků a záložek se jmény listů sešitu.

*Tabulka 14:* Struktura záznamu typu **WINDOW1**

| Pozice (offset) | Velikost (byte) | Obsah | |
|---|---|---|---|
| 4 | 2 | Vodorovná pozice okna v 1/20 obrazového bodu | |
| 6 | 2 | Svislá pozice okna v 1/20 obrazového bodu | |
| 8 | 2 | Šířka okna v 1/20 obrazového bodu | |
| 10 | 2 | Výška okna v 1/20 obrazového bodu | |
| 12 | 2 | **Bity** | **Obsah** |
| | | 0 | $0_B$ – okno je viditelné<br>$1_B$ – okno je skryté |
| | | 1 | $0_B$ – okno je otevřené<br>$1_B$ – okno je minimalizované |
| | | 3 | $0_B$ – vodorovný posuvník je skrytý<br>$1_B$ – vodorovný posuvník je viditelný |
| | | 4 | $0_B$ – svislý posuvník je skrytý<br>$1_B$ – svislý posuvník je viditelný |
| | | 5 | $0_B$ – záložky se jmény listů sešitu jsou skryté<br>$1_B$ – záložky se jmény listů sešitu jsou viditelné |
| 14 | 2 | Index právě zobrazeného listu sešitu (počátek v nule) | |
| 16 | 2 | Index první viditelné záložky se jménem listu sešitu (počátek v nule) | |
| 18 | 2 | Počet vybraných záložek se jménem listu sešitu | |
| 20 | 2 | Šířka prostoru záložek se jmény listů sešitu v 1/1000 šířky okna | |

Záznam typu **WINDOW2** (viz tabulka 15) se nachází v každém Sheet Substream a obsahuje dodatečné údaje o nastavení okna konkrétního listu sešitu. Při exportu je tento záznam zapsán s takovými nastaveními, aby se zobrazovaly výsledky vzorců, mřížka tabulky, popisky sloupců a řádků a nulové hodnoty jako prázdné buňky.

*Tabulka 15:* Struktura záznamu typu **WINDOW2**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** | |
|---|---|---|---|
| 4 | 2 | **Bity** | **Obsah** |
| | | 0 | $0_B$ – zobrazit výsledky vzorců<br>$1_B$ – zobrazit vzorce |
| | | 1 | $0_B$ – skrytá mřížka<br>$1_B$ – viditelná mřížka |
| | | 2 | $0_B$ – skryté popisky sloupců a řádků<br>$1_B$ – viditelné popisky sloupců a řádků |
| | | 4 | $0_B$ – zobrazit nulové hodnoty jako prázdné buňky<br>$1_B$ – zobrazit nulové hodnoty |
| 6 | 2 | Index prvního viditelného řádku | |
| 8 | 2 | Index prvního viditelného sloupce | |
| 10 | 4 | Index do palety barev pro barvu mřížky | |
| 14 | 2 | Hodnota zvětšení v náhledu v procentech | |
| 16 | 2 | Hodnota zvětšení v pracovním pohledu v procentech | |
| 18 | 4 | Nepoužito | |

Záznam typu **DATEMODE** (viz tabulka 16) se nachází ve Workbook Globals Substream a obsahuje základní datum pro zobrazování hodnot data a času. Všechna data jsou uložena jako počet dnů od tohoto základního data.

*Tabulka 16:* Struktura záznamu typu **DATEMODE**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** |
|---|---|---|
| 4 | 2 | Hodnota základního data:<br>$00_H$ – základním datem je 30.12.1899<br>$01_H$ – základním datem je 01.01.1904 |

Jelikož je základní datum aplikace SmartSheet, nabývající hodnoty 01.01.1970, odlišné od základního data aplikace Microsoft Excel, je během importu na základě informací z tohoto záznamu prováděn přepočet hodnot data a času v jednotlivých buňkách. Při exportu jsou hodnoty data a času přepočítávány pro základní datum 30.12.1899.

### 3.3.6 Záznamy obsahující údaje o formátech

Všechny následující typy záznamů se nacházejí ve Workbook Globals Substream.

Záznam typu **FONT** (viz tabulka 17) obsahuje údaje o fontu. Všechny záznamy tohoto typu dohromady tvoří tabulku fontů, přičemž indexy záznamů v této tabulce mají počátek v nule a odpovídají pořadí záznamů v souboru. Index 4 není používán, proto pátý záznam v souboru má index 5 namísto 4.

*Tabulka 17:* Struktura záznamu typu **FONT**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** | |
|---|---|---|---|
| 4 | 2 | Výška fontu v 1/20 obrazového bodu | |
| 6 | 2 | **Bity** | **Obsah** |
| | | 1 | $1_B$ – kurziva |
| 8 | 2 | Index do palety barev pro barvu fontu | |
| 10 | 2 | Šířka fontu:<br>$0190_H$ – normální písmo, $02BC_H$ – tučné písmo | |
| 12 | 2 | Pozice znaků:<br>$00_H$ – normální, $01_H$ – horní index, $02_H$ – dolní index | |
| 14 | 1 | Styl podtržení:<br>$00_H$ – žádné, $01_H$ – jednoduché, $02_H$ – dvojité | |
| 15 | 1 | Rodina písma | |
| 16 | 1 | Znaková sada | |
| 17 | 1 | Nepoužito | |
| 18 | proměnná | Jméno fontu (Unicode řetězec s 8bitovou délkou, viz 3.3.2) | |

Aplikace SmartSheet podporuje pouze normální pozici znaků a jednoduché podtržení, proto jsou všechny typy podtržení, kromě žádného, brány jako jednoduché. Při importu jsou rodina písma, znaková sada a jméno fontu přehlíženy a během exportu je jako jméno fontu použito Arial, znaková sada nastavena na základní systémovou ($01_H$) a rodina písma na neznámou ($00_H$).

Záznam typu **FORMAT** (viz tabulka 18) obsahuje formátovací řetězec určující způsob zobrazení hodnot buněk. Všechny záznamy tohoto typu náleží do tabulky formátovacích řetězců, přičemž záznamy v souboru představují uživatelem definované formátovací řetězce, ukládané do tabulky od indexu 164. Předchozí indexy, počínaje nulou, náleží přednastaveným formátovacím řetězcům, které se do souboru neukládají.

*Tabulka 18:* Struktura záznamu typu **FORMAT**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** |
|---|---|---|
| 4 | 2 | Index formátovacího řetězce v rámci tabulky formátovacích řetězců |
| 6 | proměnná | Formátovací řetězec (Unicode řetězec s 16bitovou délkou, viz 3.3.2) |

Popis formátovacích řetězců přesahuje rozsah tohoto textu, jelikož poskytují velké množství možností zobrazení hodnot buněk, dalece přesahující možnosti aplikace SmartSheet. Při importu se aplikace v případě komplikovaných formátovacích řetězců snaží rozpoznat nejvhodnější výsledný formát zobrazení, což může vést i ke zcela odlišnému způsobu zobrazení hodnot.

Záznam typu **XF** (viz tabulka 19) obsahuje formátovací údaje buněk a stylů. Styly nejsou aplikací SmartSheet podporovány, ale k nim náležící záznamy jsou brány jako formáty buněk, jelikož všechny záznamy tohoto typu jsou součástí tabulky formátů. Indexy záznamů, s počátkem v nule, jsou určeny jejich pořadím v souboru. Záznam s indexem 0 představuje formát základního stylu sešitu, po něm následuje 14 záznamů s formáty dalších přednastavených stylů. Na indexu 15 se nachází základní formát buňky, použitý pro buňky s hodnotou a současně bez uživatelem zadaného formátu. Nakonec následuje 5 záznamů s formáty dalších přednastavených stylů. Těchto 21 záznamů je v souboru vždy přítomno.

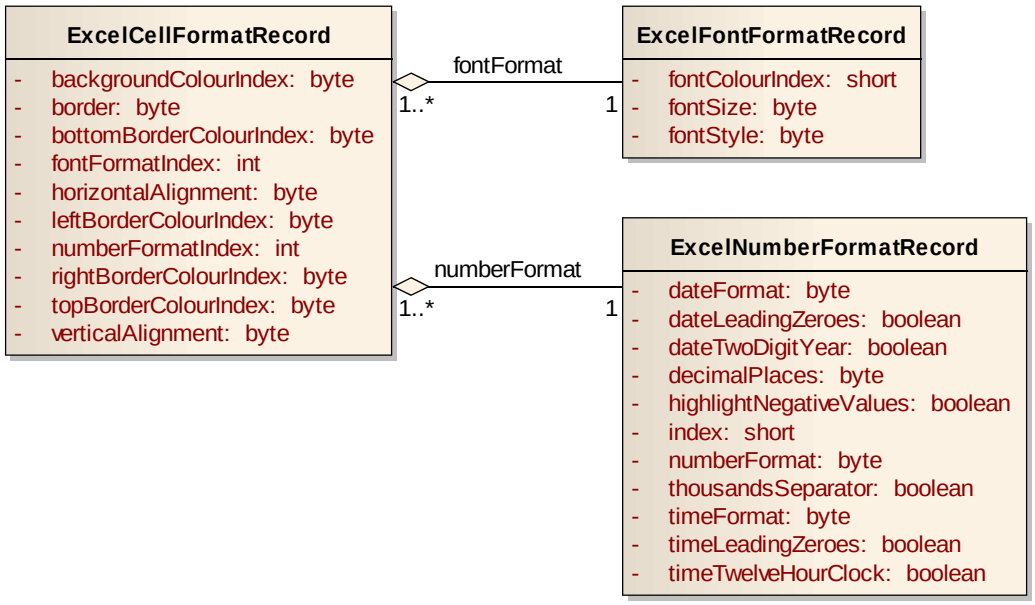
*Tabulka 19:* Struktura záznamu typu **XF**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** | |
|---|---|---|---|
| 4 | 2 | Index do tabulky fontů | |
| 6 | 2 | Index do tabulky formátovacích řetězců | |
| 8 | 2 | **Bity** | **Obsah** |
| | | 2 | $0_B$ – formát buňky, $1_B$ – formát stylu |
| | | 15-4 | Index do tabulky formátů pro formát rodičovského stylu (vždy $0FFF_H$ pro formáty stylů) |
| 10 | 2 | 2-0 | Vodorovné zarovnání:<br>$00_H$ – obecné, $01_H$ – vlevo, $02_H$ – uprostřed, $03_H$ – vpravo |
| | | 6-4 | Svislé zarovnání:<br>$00_H$ – nahoru, $01_H$ – uprostřed, $02_H$ – dolů |
| | | 15-8 | Rotace textu ve stupních |
| 12 | 1 | 3-0 | Úroveň odsazení textu vzhledem k vodorovnému zarovnání |
| | | 4 | $1_B$ – přizpůsobení velikosti textu šířce buňky |
| 13 | 1 | Příznaky použitých atributů formátu (používá pouze Microsoft Excel) | |
| 14 | 2 | **Bity** | **Obsah** |
| | | 3-0 | Styl levého okraje |
| | | 7-4 | Styl pravého okraje |
| | | 11-8 | Styl horního okraje |
| | | 15-12 | Styl dolního okraje |
| 16 | 8 | 6-0 | Index do palety barev pro barvu levého okraje |
| | | 13-7 | Index do palety barev pro barvu pravého okraje |
| | | 22-16 | Index do palety barev pro barvu horního okraje |
| | | 29-23 | Index do palety barev pro barvu dolního okraje |
| | | 47-42 | Vzor výplně |
| | | 54-48 | Index do palety barev pro barvu výplně |
| | | 61-55 | Index do palety barev pro barvu pozadí výplně |

Během importu, jakýkoliv styl okraje, kromě žádného ($00_H$), znamená že bude zobrazen okraj buňky. Stejně tak pokud je vzor výplně nastaven jako plný ($01_H$), bude mít buňka barvu pozadí odpovídající barvě výplně, jinak bude mít pozadí buňky základní barvu. Pokud má buňka zobrazen okraj, bude při exportu zapsán tenký typ okraje ($01_H$) a v případě, že má buňka jinou barvu pozadí nežli základní, bude zapsán plný vzor výplně ($01_H$), v opačném případě bude formát bez výplně ($00_H$).

### 3.3.7 Vlastní řešení čtení a zápisu záznamů s údaji o formátech

Informace o formátování buněk jsou uchovávány v souboru pomocí tří typů záznamů. Ve výsledku se každý záznam obsahující informace o vzhledu buňky odkazuje na záznam obsahující informace o fontu a na záznam s formátem zobrazení hodnoty buňky. Aplikace SmartSheet ukládá veškeré údaje o vzhledu buněk a jejich obsahu do instancí jediné třídy, a proto je nutné při importu informace z těchto tří typů záznamů sloučit a při exportu je naopak mezi tyto záznamy rozdělit. Pro účely importu slouží třídy **ExcelFontFormatRecord**, **ExcelNumberFormatRecord** a **ExcelCellFormatRecord** (viz obrázek 21).




*Obrázek 21:* Diagram tříd použitých při čtení a zápisu záznamů s údaji o formátování

Během importu je nutné brát v potaz skutečnost, že jednotlivé typy záznamu nemusejí být v souboru uloženy v ideálním pořadí, to znamená všechny záznamy typu **FONT** a **FORMAT** nemusejí být zapsány před všemi záznamy typu **XF**, které se na ně odkazují. Proto je nutné informace z jednotlivých záznamů uchovávat v paměti až do přečtení celého Workbook Globals Substream. Teprve poté je možné formátovací údaje sloučit do instancí třídy **Format** (viz 2.2.5).

Při exportu je nutné nejdříve sestavit dvě tabulky. První bude obsahovat záznamy z tabulky formátů (viz 2.4), které mají navzájem odlišné údaje týkající se fontu. Druhá tabulka bude obdobně obsahovat záznamy s odlišnými údaji o formátu zobrazení hodnot. Toho se dosáhne postupným porovnáním údajů všech záznamů tabulky formátů s údaji záznamů již obsažených ve vytvářených tabulkách. Pokud cílová tabulka neobsahuje záznam se shodnými údaji, je porovnávaný záznam přidán do cílové tabulky, v opačném případě je záznam přehlédnut. Po sestavení obou tabulek je jejich obsah zapsán do souboru ve formě záznamů typu **FONT** a **FORMAT**. Poté jsou zapisovány

záznamy typu **XF**, přičemž indexy do tabulky fontů a do tabulky formátů zobrazení hodnot se získávají porovnáním právě zapisovaného záznamu tabulky formátů se záznamy v obou tabulkách. Použijí se indexy těch záznamů, které mají shodné patřičné atributy.

### 3.3.8 Záznamy obsahující údaje o vzhledu sloupců a řádků

Všechny následující typy záznamů se nacházejí v Sheet Substream.

Záznam typu **DEFAULTCOLUMNWIDTH** (viz tabulka 20) určuje základní šířku všech sloupců v listu sešitu. Hodnota základní šířky nebude použita pro sloupce, které mají zadanou šířku přímo uživatelem.

*Tabulka 20:* Struktura záznamu typu **DEFAULTCOLUMNWIDTH**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | 2 | Základní šířka sloupce ve znacích (výsledná šířka se získá vynásobením této hodnoty šířkou znaku představujícího nulu při použití základního fontu sešitu) |

Záznam typu **COLUMNINFO** (viz tabulka 21) určuje šířku sloupce a základní formát buněk pro zadaný rozsah sloupců. Sloupce pro které tento záznam není přítomen, mají základní šířku, zadanou v záznamu typu **DEFAULTCOLUMNWIDTH**. Uvedený základní formát buněk se použije pro buňky v daném rozsahu sloupců, které jsou prázdné a není pro ně přítomen vlastní záznam.

*Tabulka 21:* Struktura záznamu typu **COLUMNINFO**

| Pozice (offset) | Velikost (byte) | Obsah | |
|---|---|---|---|
| 4 | 2 | Index prvního sloupce rozsahu | |
| 6 | 2 | Index posledního sloupce rozsahu | |
| 8 | 2 | Šířka sloupců v 1/256 šířky znaku představujícího nulu při použití základního fontu sešitu | |
| 10 | 2 | Index do tabulky formátů pro základní formát sloupců | |
| 12 | 2 | **Bity** | **Obsah** |
| | | 0 | $1_B$ – sloupce jsou skryté |
| 14 | 2 | Nepoužito | |

Při importu jsou index základního formátu i další nastavení, kromě šířky, přehlíženy, protože aplikace SmartSheet umožňuje sloupcům nastavit pouze vlastní šířku. Při exportu je pro každý sloupec, který má uživatelem nastavenou šířku, zapsán samostatný záznam, a to i v případě shodné šířky několika sousedících sloupců.

Záznam typu **DEFAULTROWHEIGHT** (viz tabulka 22) určuje základní výšku všech řádků v listu sešitu. Pokud má řádek výšku zadanou přímo uživatelem, hodnota základní výšky se pro daný řádek nepoužije.

*Tabulka 22:* Struktura záznamu typu **DEFAULTROWHEIGHT**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** | |
|---|---|---|---|
| 4 | 2 | **Bity** | **Obsah** |
| | | 0 | $1_B$ – základní výška řádku a výška základního fontu se neshodují |
| | | 1 | $1_B$ – všechny řádky se základní výškou jsou skryté |
| 6 | 2 | Základní výška řádku v 1/20 obrazového bodu | |

Při exportu je bit informující o neshodě základní výšky řádku a výšky základního fontu nastaven na hodnotu $1_B$, jinak by byla namísto nastavené základní výšky použita výška základního fontu.

Záznam typu **ROW** (viz tabulka 23) popisuje jeden řádek v listu sešitu. Pro každý řádek, který má uživatelem nastavený vzhled nebo obsahuje alespoň jednu použitou buňku, by měl být přítomen záznam. Řádky pro které tento záznam není přítomen, mají základní výšku, zadanou v záznamu typu **DEFAULTROWHEIGHT**.

*Tabulka 23:* Struktura záznamu typu **ROW**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** | |
|---|---|---|---|
| 4 | 2 | Index řádku | |
| 6 | 2 | Index prvního použitého sloupce v daném řádku | |
| 8 | 2 | Index posledního použitého sloupce v daném řádku, zvýšený o jedna | |
| 10 | 2 | **Bity** | **Obsah** |
| | | 14-0 | Výška řádku v 1/20 obrazového bodu |
| | | 15 | $0_B$ – řádek má uživatelem nastavenou výšku<br>$1_B$ – řádek má základní výšku |
| 12 | 4 | Nepoužito | |
| 16 | 2 | **Bity** | **Obsah** |
| | | 5 | $1_B$ – řádek je skrytý |
| | | 6 | $1_B$ – výška řádku a výška základního fontu se neshodují |
| | | 7 | $1_B$ – řádek má uživatelem nastaven základní formát |
| 18 | 2 | Index do tabulky formátů pro základní formát řádku | |

Při importu je brán v potaz pouze údaj o výšce, jelikož je to jediná vlastnost, kterou lze řádkům v rámci aplikace SmartSheet nastavit. Pokud má řádek uživatelem nastavenou výšku, je během exportu bit informující o neshodě mezi výškou řádku a výškou základního fontu nastaven na hodnotu $1_B$, jelikož v opačném případě by se použila výška základního fontu bez ohledu na výšku zadanou uživatelem.

### 3.3.9 Záznam reprezentující sdílenou tabulku řetězců znaků

Záznam typu **SST** (viz tabulka 24), nacházející se ve Workbook Globals Substream, obsahuje seznam všech řetězců znaků použitých v dokumentu. Každý řetězec se v tabulce vyskytuje pouze jednou. Jednotlivé řetězce v tabulce mají index s počátkem v nule, pomocí něhož se buňky, obsahující řetězcové hodnoty, do tabulky odkazují.

*Tabulka 24:* Struktura záznamu typu **SST**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** |
|---|---|---|
| 4 | 4 | Celkový počet řetězců v dokumentu |
| 8 | 4 | Počet následujících položek sdílené tabulky řetězců |
| 12 | proměnná | Seznam řetězců (Unicode řetězce s 16bitovou délkou, viz 3.3.2) |

### 3.3.10 Záznam obsahující informace o sloučených buňkách

Záznam typu **MERGEDCELLS** (viz tabulka 25) obsahuje seznam sloučených buněk pro Sheet Substream, v němž se nachází.

*Tabulka 25:* Struktura záznamu typu **MERGEDCELLS**

<table>
<tr><th>Pozice (offset)</th><th>Velikost (byte)</th><th colspan="2">Obsah</th></tr>
<tr><td>4</td><td>2</td><td colspan="2">Počet následujících položek seznamu sloučených buněk (cnt)</td></tr>
<tr><td rowspan="5">6</td><td rowspan="5">8·cnt</td><td>Bity</td><td>Obsah</td></tr>
<tr><td>15-0</td><td>Index prvního řádku</td></tr>
<tr><td>31-16</td><td>Index posledního řádku</td></tr>
<tr><td>47-32</td><td>Index prvního sloupce</td></tr>
<tr><td>63-48</td><td>Index posledního sloupce</td></tr>
</table>

Pokud je velikost seznamu sloučených buněk větší než maximální velikost záznamu, není zbytek seznamu zapsán v záznamech typu **CONTINUE**, ale je zapsán v jednom nebo více dalších samostatných záznamech typu **MERGEDCELLS**.

### 3.3.11 Záznamy představující buňky s konstantními hodnotami

Všechny následující typy záznamů se nacházejí v Sheet Substream. Každý záznam obsahuje indexy řádku a sloupce s počátkem v nule, a také index do tabulky formátů, viz 3.3.6

Záznam typu **BLANK** (viz tabulka 26) představuje prázdnou buňku, která neobsahuje žádnou hodnotu, ale má uživatelem nastaven formát.

*Tabulka 26:* Struktura záznamu typu **BLANK**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | 2 | Index řádku |
| 6 | 2 | Index sloupce |
| 8 | 2 | Index do tabulky formátů |

Záznam typu **MULTIPLEBLANK** (viz tabulka 27) reprezentuje více prázdných buněk, které jsou umístěny vedle sebe ve stejném řádku. Při exportu není tento typ záznamy využíván.

*Tabulka 27:* Struktura záznamu typu **MULTIPLEBLANK**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | 2 | Index řádku |
| 6 | 2 | Index prvního sloupce (**fc**) |
| 8 | 2·(**lc**-**fc**+1) | Seznam indexů do tabulky formátů |
| proměnná | 2 | Index posledního sloupce (**lc**) |

Záznam typu **BOOLEANERROR** (viz tabulka 28) představuje buňku s pravdivostní nebo chybovou hodnotou.

*Tabulka 28:* Struktura záznamu typu **BOOLEANERROR**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | 2 | Index řádku |
| 6 | 2 | Index sloupce |
| 8 | 2 | Index do tabulky formátů |
| 10 | 1 | Pokud je typem hodnoty pravdivostní hodnota:<br>$00_H$ – pravda, $01_H$ – nepravda<br>Pokud je typem hodnoty chybová hodnota:<br>$00_H$ – prázdný průnik rozsahů buněk<br>$07_H$ – dělení nulou<br>$0F_H$ – neočekávaný typ hodnoty<br>$17_H$ – neplatný odkaz<br>$1D_H$ – neznámé označení<br>$24_H$ – překročen rozsah hodnoty<br>$2A_H$ – nedostupná hodnota |
| 11 | 1 | Typ hodnoty:<br>$00_H$ – pravdivostní hodnota, $01_H$ – chybová hodnota |

Záznam typu **LABELSST** (viz tabulka 29) reprezentuje buňky obsahující řetězec znaků. Buňka samotná řetězec neobsahuje, pouze se odkazuje pomocí indexu do sdílené tabulky řetězců znaků, viz 3.3.9.

*Tabulka 29:* Struktura záznamu typu **LABELSST**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | 2 | Index řádku |
| 6 | 2 | Index sloupce |
| 8 | 2 | Index do tabulky formátů |
| 10 | 4 | Index do sdílené tabulky řetězců znaků |

Záznam typu **NUMBER** (viz tabulka 30) představuje buňku obsahující číselnou hodnotu ve formátu IEEE 754.

*Tabulka 30:* Struktura záznamu typu **NUMBER**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | 2 | Index řádku |
| 6 | 2 | Index sloupce |
| 8 | 2 | Index do tabulky formátů |
| 10 | 8 | Číselná hodnota ve formátu IEEE 754 |

Následující dva typy záznamů ukládají číselné hodnoty ve formátu RK (viz tabulka 31).

*Tabulka 31:* Přiřazení bitů v rámci číselného formátu RK

| Bity | Obsah |
|---|---|
| 0 | $1_B$ – zakódovaná hodnota se musí podělit 100 |
| 1 | $0_B$ – zakódovaná hodnota představuje 30 nejvyšších bitů číselné hodnoty ve formátu IEEE 754, nejnižších 34 bitů musí být nastaveno na nulu<br>$1_B$ – zakódovaná hodnota představuje 30bitovou celočíselnou hodnotu |
| 31-2 | Zakódovaná číselná hodnota |

Záznam typu **RK** (viz tabulka 32) reprezentuje buňku s číselnou hodnotou. Při exportu není tento typ záznamu využíván, vždy je použit záznam typu **NUMBER**.

*Tabulka 32:* Struktura záznamu typu **RK**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | 2 | Index řádku |
| 6 | 2 | Index sloupce |
| 8 | 2 | Index do tabulky formátů |
| 10 | 4 | Číselná hodnota ve formátu RK |

Záznam typu **MULTIPLERK** (viz tabulka 33) reprezentuje více buněk obsahujících číselnou hodnotu, které jsou umístěny vedle sebe ve stejném řádku. Při exportu není tento typ záznamu využíván.

*Tabulka 33:* Struktura záznamu typu **MULTIPLERK**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** | |
|---|---|---|---|
| 4 | 2 | Index řádku | |
| 6 | 2 | Index prvního sloupce (**fc**) | |
| 8 | 6·(**lc**-**fc**+1) | **Velikost (byte)** | **Obsah** |
| | | 2 | Index do tabulky formátů |
| | | 4 | Číselná hodnota ve formátu RK |
| proměnná | 2 | Index posledního sloupce (**lc**) | |

### 3.3.12 Záznamy představující buňky se vzorci

Všechny následující typy záznamů se nacházejí v Sheet Substream. Některé záznamy obsahují indexy řádku a sloupce s počátkem v nule, a také index do tabulky formátů, viz 3.3.6

Záznam typu **FORMULA** (viz tabulka 34) představuje buňku obsahující vzorec. Záznam obsahuje také poslední vypočítaný výsledek vzorce, pokud není výsledkem řetězec znaků.

*Tabulka 34:* Struktura záznamu typu **FORMULA**

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** | |
|---|---|---|---|
| 4 | 2 | Index řádku | |
| 6 | 2 | Index sloupce | |
| 8 | 2 | Index do tabulky formátů | |
| Pokud je výsledkem vzorce číselná hodnota | | | |
| 10 | 8 | Číselná hodnota ve formátu IEEE 754 | |
| Pokud je výsledkem vzorce řetězec znaků, pravdivostní nebo chybová hodnota | | | |
| 10 | 8 | **Bity** | **Obsah** |
| | | 7-0 | $00_H$ – řetězec, $01_H$ – pravdivostní hodnota, $02_H$ - chyba |
| | | 15-8 | Nepoužito |
| | | 23-16 | V případě řetězce znaků nepoužito, jinak hodnota jako v záznamu typu BOOLEANERROR (viz 3.3.11) |
| | | 47-24 | Nepoužito |
| | | 63-48 | $FFFF_H$ |
| 18 | 2 | **Bity** | **Obsah** |
| | | 0 | $1_B$ – vždy přepočítat |
| | | 1 | $1_B$ – přepočítat při otevření souboru |
| 20 | 4 | Nepoužito | |
| 24 | proměnná | Data vzorce, viz 3.3.13 | |

Při exportu je vždy pro vzorec nastaveno přepočítání při otevření souboru. Pokud je výsledkem vzorce řetězec znaků, je výsledek uložen v záznamu typu **STRING** (viz tabulka 35), který se nachází hned za patřičným záznamem typu **FORMULA**.

*Tabulka 35:* Struktura záznamu typu **STRING**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | proměnná | Výsledek vzorce (Unicode řetězec s 16bitovou délkou, viz 3.3.2) |

Pro úsporu místa jsou aplikací Microsoft Excel využívány záznamy typu **SHAREDFORMULA** (viz tabulka 36), které reprezentují souvislou oblast buněk obsahujících vzorec. Ty vznikají například zkopírováním buňky se vzorcem do sousedních buněk.

*Tabulka 36:* Struktura záznamu typu **SHAREDFORMULA**

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 4 | 2 | Index prvního řádku |
| 6 | 2 | Index posledního řádku |
| 8 | 1 | Index prvního sloupce |
| 9 | 1 | Index posledního sloupce |
| 10 | 2 | Nepoužito |
| 12 | proměnná | Data vzorce, viz 3.3.13 |

Vzorec obsažený v tomto záznamu náleží buňce v prvním řádku a sloupci zadaného rozsahu, do ostatních buněk je nakopírován a v něm obsažené relativní odkazy mají své cíle posunuty o vzdálenost mezi buňkou v jejímž vzorci jsou umístěny a buňkou v prvním řádku a sloupci zadaného rozsahu.

### 3.3.13 Vzorce

Vzorce nejsou ukládány v podobě, ve které byly uživatelem zadány, ale jako pole symbolů ve formátu RPN (postfixová notace), kdy operátor následuje své operandy a výpočet probíhá za použití zásobníku. Například výraz **2 + (4 · 5)** má v RPN podobu **2 4 5 · ( ) +**. Stejné pořadí mají také symboly tvořící takový vzorec. Celková data vzorce mají podobu znázorněnou v tabulce 37.

*Tabulka 37:* Struktura dat vzorce

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 0 | 2 | Velikost následujícího pole symbolů vzorce (**sz**) |
| 2 | **sz** | Pole symbolů vzorce (RPN) |
| 2+**sz** | proměnná | Pole záznamů s dodatečným obsahem některých symbolů |

Některé symboly neukládají všechny své informace v sobě, ale v poli záznamů s dodatečným obsahem, nacházejícím se za samotným polem symbolů. Pořadí záznamů v tomto poli se shoduje s pořadím výskytu přidružených symbolů v poli symbolů.

Obsah symbolů je vždy tvořen identifikátorem, po němž v některých případech následuje další obsah symbolu (viz tabulka 38). Velikost tohoto obsahu není v rámci symbolu uložena, odvíjí se od identifikátoru.

*Tabulka 38:* Struktura symbolu

| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 0 | 1 | Identifikátor symbolu |
| 1 | proměnná | Obsah symbolu |

Symboly unárních a binárních operátorů jsou tvořeny pouze svým identifikátorem. Seznam operátorů je v tabulce 39.

*Tabulka 39:* Unární a binární operátory

| Typ | Operátor |
|---|---|
| Unární operátory | Znaménko plus (+), znaménko minus (-), procenta (%) |
| Binární operátory | Sčítání (+), odčítání (-), násobení (*), dělení (/), umocnění (^), spojení řetězců (&)<br>Menší (<), menší nebo rovná se (<=), rovná se (=), větší nebo rovná se (>=), větší (>), nerovná se (<>)<br>Průnik rozsahů buněk (mezera), sjednocení rozsahů buněk (čárka nebo středník), rozsah buněk (:) |

Symboly konstantních operandů obsahují kromě identifikátoru také informace o hodnotě (viz tabulka 40).

*Tabulka 40:* Konstantní operandy

| Operand | Velikost informace o hodnotě (byte) | Obsah |
|---|---|---|
| Chybějící parametr funkce | 0 | Symbol nenese žádný další obsah, pouze svůj identifikátor (operand je při výpočtech považován za nulovou hodnotu) |
| Řetězec znaků | proměnná | Obsah řetězce znaků (Unicode řetězec s 8bitovou délkou, viz 3.3.2) |
| Chybová konstanta | 1 | Chybová hodnota jako v záznamu typu BOOLEANERROR (viz 3.3.11) |
| Pravdivostní konstanta | 1 | Pravdivostní hodnota jako v záznamu typu BOOLEANERROR (viz 3.3.11) |
| Celočíselná konstanta | 2 | Kladná celočíselná hodnota |
| Reálná konstanta | 8 | Číselná hodnota ve formátu IEEE 754 |

Symboly proměnných operandů jsou tvořeny adresami jednotlivých buněk a adresami rozsahů buněk. Symbol adresy buňky tak kromě identifikátoru obsahuje také indexy sloupce a řádku,

včetně informace o tom, zda jsou relativní či nikoliv (viz tabulka 41).

*Tabulka 41:* Operand adresy buňky

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** | |
|---|---|---|---|
| 1 | 2 | Index řádku adresy | |
| 3 | 2 | **Bity** | **Obsah** |
| | | 13-0 | Index sloupce adresy |
| | | 14 | $1_B$ – adresa sloupce je relativní |
| | | 15 | $1_B$ – adresa řádku je relativní |

Symbol rozsahu buněk reprezentuje odkaz na obdélníkovou oblast v tabulce a obsahuje, kromě identifikátoru, indexy prvních a posledních sloupců a řádků, včetně informace o jejich relativnosti (viz tabulka 42).

*Tabulka 42:* Operand rozsahu buněk

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** | |
|---|---|---|---|
| 1 | 2 | Index prvního řádku rozsahu | |
| 3 | 2 | Index posledního řádku rozsahu | |
| 5 | 2 | **Bity** | **Obsah** |
| | | 13-0 | Index prvního sloupce rozsahu |
| | | 14 | $1_B$ – adresa prvního sloupce je relativní |
| | | 15 | $1_B$ – adresa prvního řádku je relativní |
| 7 | 2 | **Bity** | **Obsah** |
| | | 13-0 | Index posledního sloupce rozsahu |
| | | 14 | $1_B$ – adresa posledního sloupce je relativní |
| | | 15 | $1_B$ – adresa posledního řádku je relativní |

Symboly funkcí jsou zvláštní případy operátorů. Stejně jako symboly jiných operátorů, symboly funkcí následují v RPN vzorci své operandy (parametry). Některé funkce mají pevný počet parametrů a jejich symbol proto informaci o počtu zadaných parametrů neobsahuje, jelikož je vázána na identifikátor dané funkce (viz tabulka 43).

*Tabulka 43:* Funkce s pevným počtem parametrů

| **Pozice (offset)** | **Velikost (byte)** | **Obsah** |
|---|---|---|
| 1 | 2 | Identifikátor funkce |

Funkce s proměnným počtem parametrů obsahují mimo svůj identifikátor také informaci o počtu zadaných parametrů (viz tabulka 44).

*Tabulka 44:* Funkce s proměnným počtem parametrů

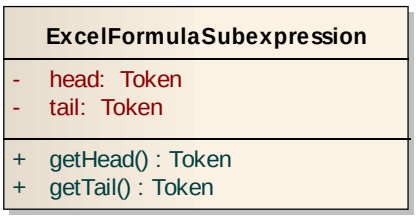
| Pozice (offset) | Velikost (byte) | Obsah |
|---|---|---|
| 1 | 1 | Počet zadaných parametrů funkce |
| 2 | 2 | Identifikátor funkce |

Aplikace SmartSheet nezná informace o počtu parametrů všech funkcí aplikace Microsoft Excel, a při importu tedy může vzniknout nesmyslný a nefunkční vzorec, pokud je v jeho originále obsažena neznámá funkce.

Ve vzorcích se mohou objevit další druhy symbolů, které jsou buďto importovány (závorky), opomíjeny (optimalizační a řídící symboly) nebo převedeny na chybovou hodnotu (zvláštní druhy adres).

### 3.3.14 Vlastní řešení čtení a zápisu vzorců

Aplikace SmartSheet neukládá vzorce v postfixové notaci, ale jako zřetězený seznam symbolů, a to v pořadí v jakém byly zadány. Při importu je proto nutné převést vzorce v postfixové notaci na zřetězený seznam symbolů. K tomu slouží zásobník, do něhož se ukládají instance třídy **ExcelFormulaSubexpression** (viz obrázek 22). Instance této třídy vždy obsahují zřetězený seznam symbolů části vzorce, přičemž atribut **head** odkazuje na počáteční symbol seznamu a atribut **tail** na koncový symbol seznamu.




*Obrázek 22:* Diagram třídy ExcelFormulaSubexpression

Import vzorce tedy probíhá obdobně jako výpočet pomocí zásobníku. Pokud je ze souboru přečten symbol operandu, je uložen jako instance výše zmíněné třídy do zásobníku, přičemž atributy **head** a **tail** odkazují na stejný symbol. Pokud je přečten symbol operátoru nebo funkce, je ze zásobníku získán potřebný počet operandů, a z těchto všech symbolů je sestaven zřetězený seznam, tvořící část vzorce, který je posléze uložen do zásobníku. Části vzorce, od jednotlivých operandů po celé výrazy, jsou takto postupně spojovány v jediný zřetězený seznam, dokud není přečten celý vzorec.

Při exportu je pro převedení do postfixové formy využito metody rekurzivního sestupu, stejně jako při výpočtu. Namísto vypočtených hodnot jsou však do zásobníku ukládány jednotlivé symboly zřetězeného seznamu. Takto vznikne reprezentace vzorce v postfixové notaci, která je posléze uložena do souboru.

# 4 Vytváření uživatelských funkcí

Tato kapitola popisuje způsob, jakým je realizováno vytváření uživatelských funkcí. Jsou zde popsány základní datové struktury a algoritmy umožňující tvorbu a použití těchto funkcí.

## 4.1 Důvod a předpoklady zavedení tvorby uživatelských funkcí

Hlavním důvodem pro přidání této funkcionality do aplikace, byla možnost nabídnout uživateli nástroj pro rozšíření knihovny výpočetních funkcí. Dalším důvodem byla možnost zapojit do výpočtu konstrukce, které nejsou v samotných vzorcích možné nebo jednoduše proveditelné, jako například podmínky, cykly a využití proměnných pro mezivýsledky. Takto vytvořené uživatelské funkce mohou také nahradit opakovaně používané a složité části vzorců, čímž lze ušetřit paměť.

Nutný předpoklad byl, aby tato tvorba uživatelských funkcí měla v co největší míře vizuální podobu, tedy bez použití jakéhokoliv textového programovacího jazyka. Důvodem byly nejenom omezené možnosti vstupu na cílové skupině mobilních zařízení, ale také nižší nároky na uživatele.

Výsledný programovací nástroj musel splňovat následující výčet vlastností:

- uživatel určuje název funkce a počet vstupních parametrů
- vstupním parametrům lze přiřadit základní hodnotu, použitou při vynechání parametru, a nastavit kolik parametrů je povinných
- během výpočtu lze použít předem vytvořené proměnné
- v programu lze použít všechny dostupné unární a binární operátory, a také všechny vestavěné i uživatelské funkce
- pro řízení průběhu výpočtu slouží konstrukce podmínky a cyklu

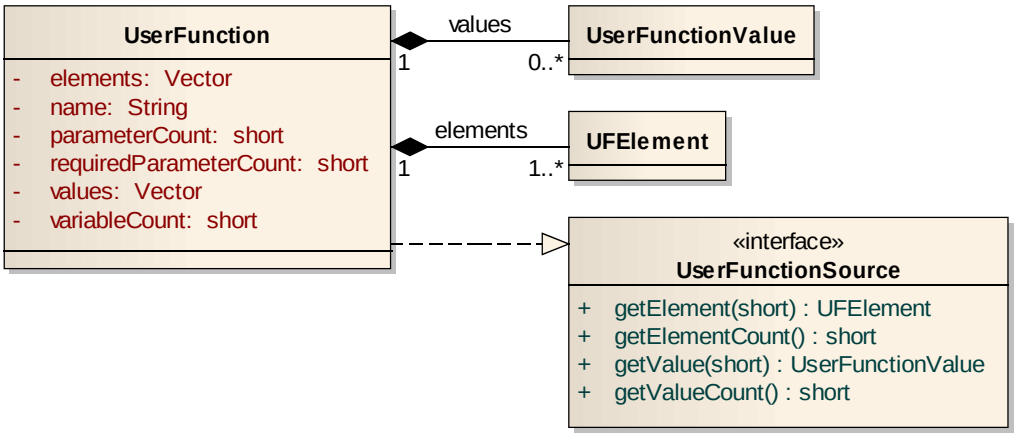
Výsledný program uživatelské funkce lze popsat jako řetězec složený z jednotlivých stavebních prvků, kdy se postupuje od prvního prvku řetězce směrem k prvnímu návratovému prvku, a po celou dobu jsou prováděny operace přidružené jednotlivým stavebním členům uživatelské funkce.

## 4.2 Reprezentace uživatelských funkcí

Samotná uživatelská funkce je reprezentována třídou **UserFunction** (viz obrázek 23), kde atribut **name** obsahuje název funkce používaný ve vzorcích. Uživatel může zadat parametry přijímané funkcí, přičemž jejich počet je vyjádřen atributem **parameterCount**. Z těchto parametrů mohou být povinné všechny nebo pouze určitý počet, což je vyjádřeno atributem **requiredParameterCount**. Povinné parametry jsou počítány zleva, tedy například funkce se třemi parametry a dvěma povinnými parametry musí mít ve vzorci první dva parametry zadané, ale poslední třetí nikoliv. Atribut **variableCount** obsahuje počet vytvořených proměnných. Vytvořené parametry i proměnné jsou uloženy ve vektoru **values** jako instance třídy **UserFunctionValue**. Vektor **elements** obsahuje jednotlivé stavební prvky uživatelské funkce v podobě instancí třídy **UFElement** a dědičných tříd. Umístění prvku v tomto vektoru určuje jeho umístění v rámci programu uživatelské funkce a výpočet je tedy započat u prvku na nultém indexu.

Třída **UserFunction** dále realizuje rozhraní **UserFunctionSource**, čímž se z ní stává zdroj hodnot pro jednotlivé prvky funkce. Metoda **getElement** vrací prvek na zadaném indexu a metoda **getElementCount** vrací počet prvků dané uživatelské funkce. Obdobně metoda **getValue** vrací

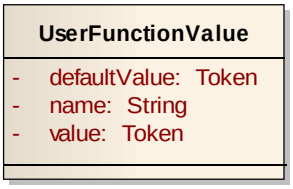
parametr nebo proměnnou na zadaném indexu a metoda **getValueCount** vrací celkový počet hodnot uživatelské funkce.




*Obrázek 23:* Diagram tříd reprezentujících uživatelskou funkci

### 4.2.1 Třída UserFunctionValue

Třída **UserFunctionValue** (viz obrázek 24) představuje parametr nebo proměnnou uživatelské funkce. Po započetí zpracování uživatelské funkce lze parametry použít stejným způsobem jako proměnné, tudíž lze do nich také zapisovat nové hodnoty.



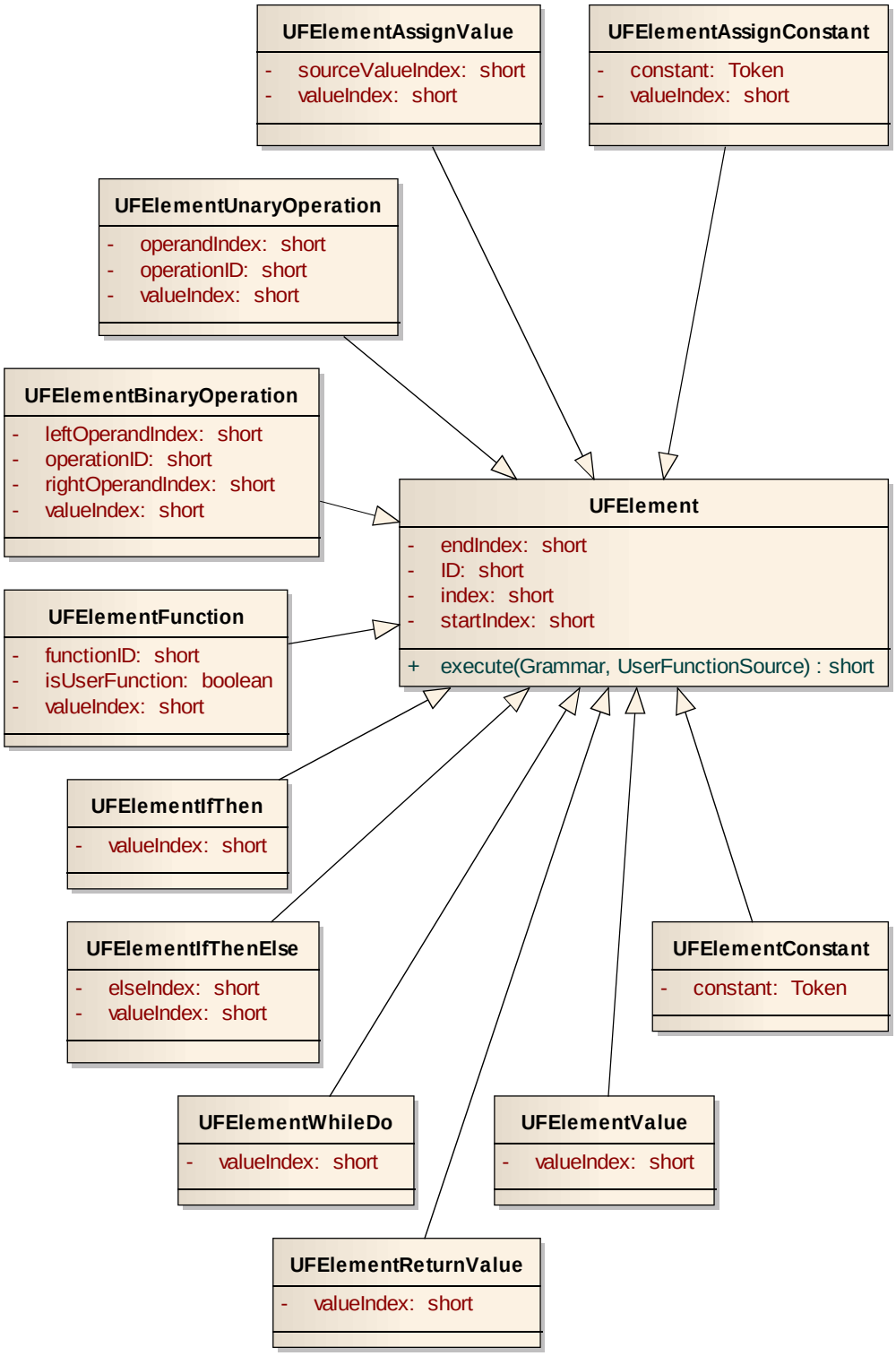

*Obrázek 24:* Diagram třídy UserFunctionValue

Atribut **name** obsahuje název hodnoty. Ten je určen pouze pro uživatele, jelikož vnitřně jsou hodnoty odkazovány pouze pomocí svých indexů v rámci vektoru **values** třídy **UserFunction**. V atributu **value** je umístěna aktuální hodnota v podobě instance třídy **Token** a dědičných tříd. Atribut **defaultValue** obsahuje základní hodnotu používanou u nepovinných parametrů. Uživatel má možnost pro nepovinné parametry nastavit základní hodnotu, která se použije v případě, že tento parametr není při volání funkce zadán. Pokud není zadán parametr ani základní hodnota, má takový parametr na počátku nulovou hodnotu.

### 4.2.2 Třída UFElement

Třída **UFElement** a její dědičné třídy tvoří hierarchii tříd reprezentujících stavební prvky uživatelské funkce. Atribut **ID** určuje typ stavebního prvku. Trojice atributů **index**, **startIndex**, **endIndex** obsahují obsahují index, počáteční index a koncový index daného prvku. Atribut **index** má vždy hodnotu pozice patřičného prvku v rámci vektoru **elements** třídy **UserFunction**. V případě prvků, které nemají přidružený žádný koncový prvek (například prvek pro přiřazení hodnoty do proměnné), mají atributy **startIndex** a **endIndex** stejnou hodnotu jako **index**. Pokud se však jedná o dvojici počátečního a koncového prvku (například podmínka nebo cyklus), pak v počátečním prvku mají atributy **index** a **startIndex** stejnou hodnotu, ale atribut **endIndex** obsahuje

pozici koncového prvku. Naopak v koncovém prvku mají atributy **index** a **endIndex** shodnou hodnotu a atribut **startIndex** obsahuje pozici počátečního prvku. Tato znalost pozice jednotlivých prvků je důležitá pro metodu **execute**, která poté co vykoná operaci spojenou s daným typem prvku, vrací index prvku na kterém má zpracování uživatelské funkce pokračovat. Celá hierarchie třídy **UFElement** a dědičných tříd je zachycena na obrázku 25.




*Obrázek 25:* Diagram hierarchie tříd reprezentujících stavební prvky uživatelské funkce

Třída **UFElementAssignValue** provádí přiřazení hodnoty proměnné s indexem **sourceValueIndex** do proměnné na indexu **valueIndex**, zatímco třída **UFElementAssignConstant** přiřazuje hodnotu **constant** do proměnné s indexem **valueIndex**. Zpracování funkce poté pokračuje na následujícím prvku.

Třída **UFElementUnaryOperation** provádí unární operaci **operationID** nad hodnotou proměnné s indexem **operandIndex** a výsledek ukládá do proměnné na indexu **valueIndex**. Obdobně pracuje třída **UFElementBinaryOperation**, která provádí binární operaci **operationID** nad hodnotami proměnných s indexy **leftOperandIndex** a **rightOperandIndex** a výsledek opět ukládá do proměnné na indexu **valueIndex**. Zpracování funkce pokračuje na následujícím prvku.

Třídy **UFElementValue** a **UFElementConstant** neprovádí žádnou činnost, ale slouží jako zdroj hodnot pro třídu **UFElementFunction**. Ta získává hodnoty ze všech prvků typu **UFElementValue**, **UFElementConstant**, **UFElementAssignValue**, **UFElementAssignConstant**, **UFElementUnaryOperation**, **UFElementBinaryOperation** a **UFElementFunction** umístěných mezi počátečním prvkem a koncovým prvkem a používá je jako parametry pro volání funkce **functionID**. Výsledek ukládá do proměnné na indexu **valueIndex**. Atribut **isUserFunction** určuje zda se jedná o uživatelskou funkci či nikoliv. Zpracování vnitřních prvků probíhá stejně jako v celém programu uživatelské funkce, takže pokud se uvnitř nachází podmínka, vyberou se jako parametry pro volání funkce pouze ty prvky, které se nacházejí v části určené splněním nebo nesplněním podmínky. Takto lze využít řídících prvků pro získávání hodnot parametrů.

Třída **UFElementIfThen** představuje podmínku. Pokud je hodnota proměnné na indexu **valueIndex** pravdivá, pokračuje zpracování funkce na prvku za počátečním prvkem podmínky, v opačném případě za koncovým prvkem podmínky. Třída **UFElementIfThenElse** doplňuje podmínku o index **elseIndex**, od kterého pokračuje zpracování funkce při nesplnění podmínky.

Třída **UFElementWhileDo** představuje cyklus, který se provádí dokud je hodnota proměnné na indexu **valueIndex** pravdivá. Při pravdivé hodnotě pokračuje zpracování na prvku za počátečním prvkem dokud se nedostane ke koncovému prvku cyklu, který odkáže zpracování zpět na počáteční prvek cyklu. V opačném případě pokračuje zpracování za koncovým prvkem cyklu.

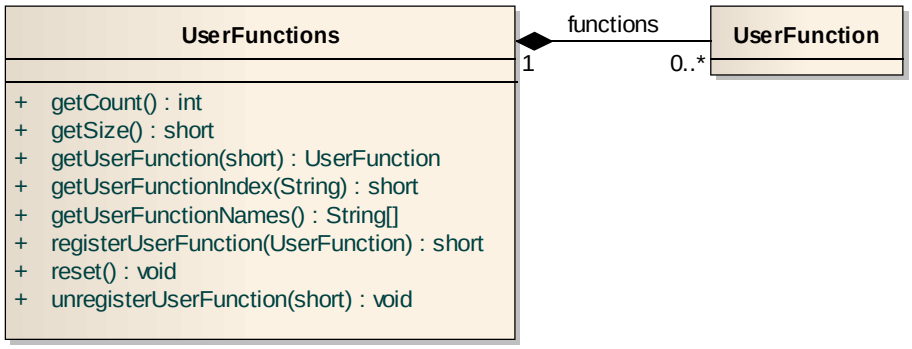
Třída **UFElementReturnValue** ukončuje zpracování funkce a vrací hodnotu proměnné na indexu **valueIndex**. Třída **UFElement** reprezentuje všechny typy koncových prvků a také zvláštní návratový prvek vracející nulovou hodnotu, který se vždy nachází na konci každé uživatelské funkce.

## 4.3 Správa uživatelských funkcí

Správa uživatelských funkcí je reprezentována třídou **UserFunctions** (viz obrázek 26), která ukládá všechny uživatelské funkce ve vektoru **functions** jako instance třídy **UserFunction** a představuje tak tabulku uživatelských funkcí.

Metoda **registerUserFunction** provádí kontrolu názvu funkce, zda neobsahuje nepovolené znaky a zda již neexistuje funkce se shodným názvem, a pokud je vše v pořádku, tak funkci přidá na konec tabulky. Naopak metoda **unregisterUserFunction** odstraňuje funkci na zadaném indexu z tabulky. Prázdná místa v tabulce jsou odstraněna až při ukládání do souboru, spolu s úpravou indexace zbylých uživatelských funkcí. Metoda **getUserFunction** vrací funkci na zadaném indexu, metoda **getUserFunctionIndex** vrací index funkce se zadaným názvem. Metoda **getUserFunctionNames** vrací pole názvů všech uživatelských funkcí.

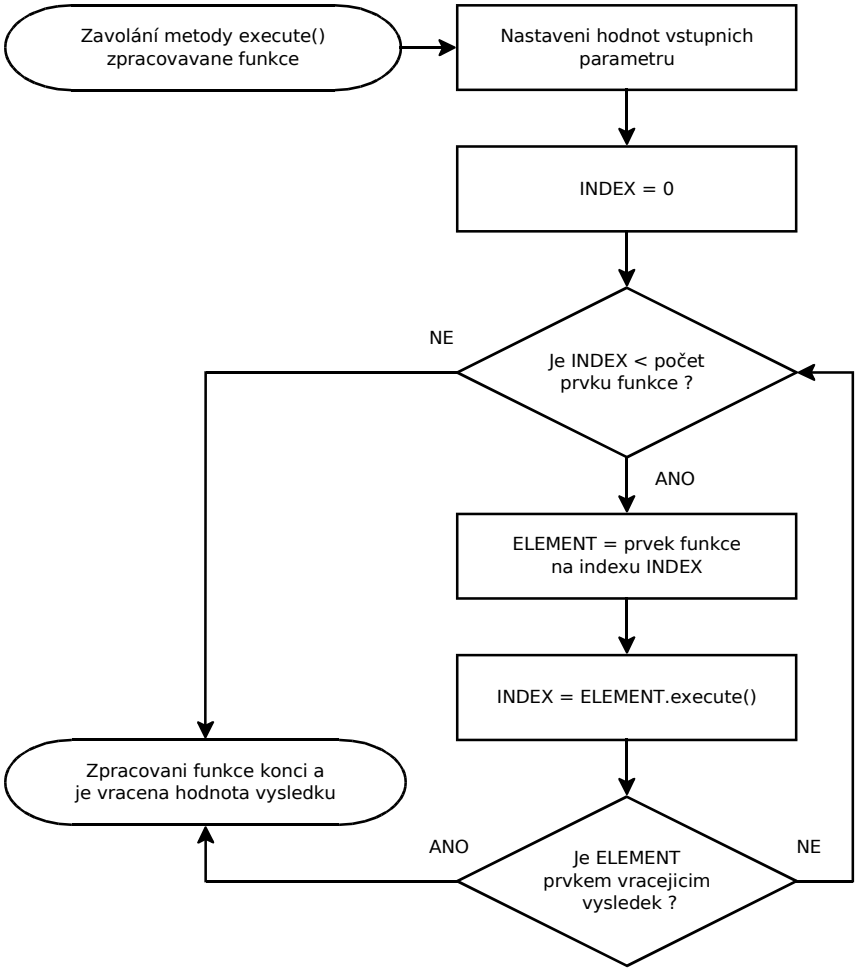
Velikost tabulky se zjistí pomocí metody **getSize**, ale v tomto případě se do počtu záznamů započítávají také prázdná místa mezi záznamy. Skutečný počet záznamů, a tedy také uživatelských funkcí, vrací metoda **getCount**. Všechny záznamy se odstraní pomocí metody **reset**.




*Obrázek 26:* Diagram tříd reprezentujících správu uživatelských funkcí

## 4.4 Průběh zpracování uživatelské funkce

Algoritmus zpracování uživatelské funkce je zachycen na obrázku 27. Před započetím zpracování jsou parametrům nastaveny hodnoty zadané jako parametry funkce ve vzorci.




*Obrázek 27:* Algoritmus zpracování uživatelské funkce

Zpracování poté začíná na prvku s nulovým indexem. Po zavolání metody **execute** právě zpracovávaného prvku je zjištěn index následujícího prvku a tam také pokračuje zpracování funkce. Toto se opakuje dokud zpracovávaným prvkem není prvek návratový, kdy je zpracování ukončeno a je vrácena hodnota výsledku funkce.

## 4.5 Sdílení uživatelských funkcí

Registrované uživatelské funkce se ukládají pouze jako součást tabulky, neexistuje tedy žádná sdílená knihovna funkcí. Pokud chce uživatel použít již vytvořenou funkci v jiné tabulce, může k tomuto účelu využít možnost importu funkcí ze souboru do aktuální tabulky. Vybrané funkce jsou přidány na konec tabulky uživatelských funkcí.

# 5 Vlastní formát souborů

V rámci této kapitoly je popsán vlastní formát souborů aplikace SmartSheet. Jsou zde popsány jeho vlastnosti a struktura uložených dat.

## 5.1 Vlastnosti formátu

Jedná se o binární formát souborů, kterému byla dána přednost před variantou založenou na XML, a to především z důvodu vyšší rychlosti a nižší paměťové náročnosti zpracování binárních dat. Výsledný soubor se skládá ze záznamů, obdobně jako je tomu například u souborového formátu BIFF8 aplikace Microsoft Excel. Struktura záznamu je popsána v tabulce 45.

*Tabulka 45:* Struktura záznamu

| **Velikost (byte)** | **Obsah** |
|---|---|
| 2 | Identifikátor záznamu |
| 2 | Velikost dat záznamu (**sz**) |
| **sz** | Data záznamu |

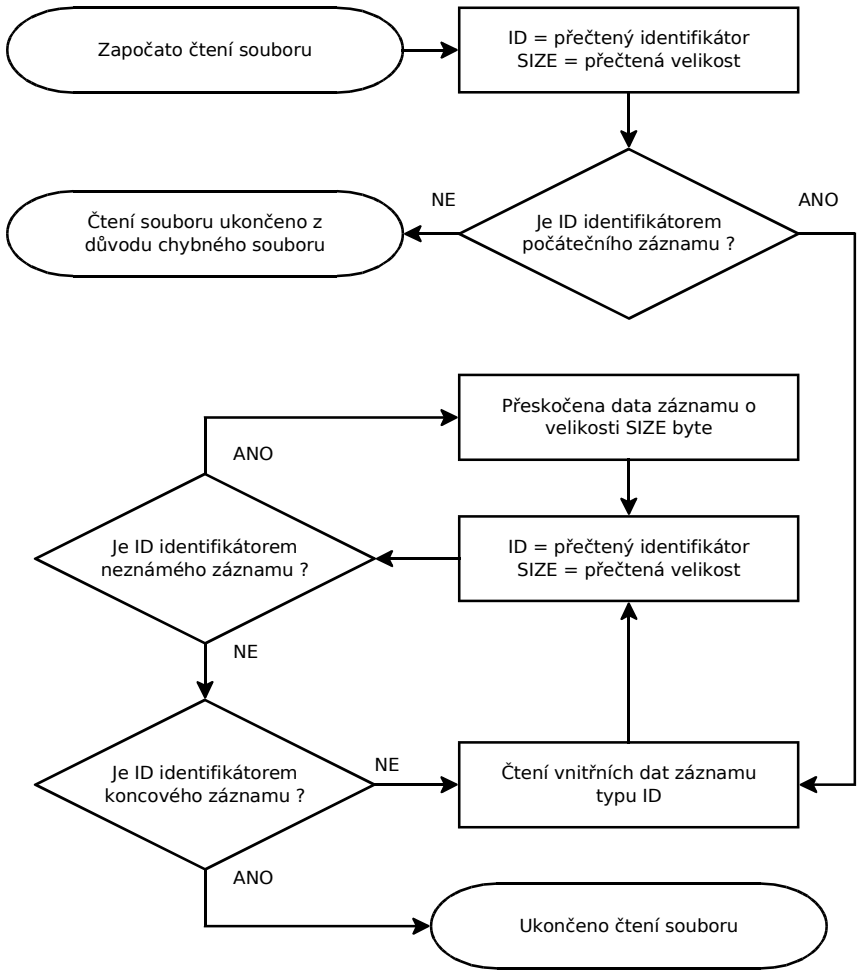
Typ záznamu je určen identifikátorem a velikost záznamu umožňuje přeskočení neznámých typů záznamů, což může být nutné při načítání souboru vytvořeného novější verzí aplikace do starší verze aplikace. Pokud tedy nedojde ke změně struktury vnitřních dat již existujících typů záznamů, lze tímto způsobem zajistit kompatibilitu novějších verzí formátu souboru se staršími verzemi aplikace. Způsob načítání souborů je zachycen na obrázku 28.

Kompatibilitu lze zachovat při rozšiřování funkcionality aplikace tak, že se nebude měnit struktura vnitřních dat záznamů, ale k těmto záznamům se budou vytvářet nové typy záznamů, které budou jejich obsah rozšiřovat. Nová verze aplikace je zpracuje, ale předchozí je budou přehlížet.

Záznamy, kromě počátečního a koncového, mohou být v souboru umístěny v libovolném pořadí. Avšak v případě záznamů obsahujících údaje do indexovaných tabulek, určuje pořadí záznamu index těchto údajů v rámci tabulky. Tedy první výskyt záznamu obsahuje údaje na nultém indexu v tabulce, druhý výskyt na prvním indexu, atd.

Hodnoty zapsané v rámci záznamů mají následující vlastnosti:

- hodnoty rozdělené do více byte jsou uloženy s uspořádáním big-endian
- řetězce znaků jsou uloženy jako pole znaků s 16bitovým Unicode kódováním, kde každý znak zabírá 2 byte.




*Obrázek 28:* Algoritmus čtení souboru

## 5.2 Zápis symbolů vzorců a hodnot buněk

Symboly z nichž jsou tvořeny vzorce a hodnoty buněk, jsou zapisovány do souboru jako pole, kde jednotlivé zapsané symboly mají strukturu vyjádřenou v tabulce 46. Symboly tvořící vzorec jsou tímto způsobem uloženy v souboru ve stejném pořadí, v jakém jsou zadány uživatelem do daného vzorce.

*Tabulka 46:* Struktura dat symbolu

| **Velikost (byte)** | **Obsah** |
|---|---|
| 2 | Identifikátor symbolu |
| proměnná | Dodatečná data symbolu |

Některé symboly mají za identifikátorem uloženy dodatečná data, jejichž struktura je popsány v tabulce 47. Pole symbolů je vždy ukončeno zvláštním koncovým symbolem, jelikož záznamy neobsahují žádné informace o počtu symbolů ve vzorci.

*Tabulka 47:* Struktura dodatečných dat symbolů

| Symbol | Velikost (byte) | Obsah |
|---|---|---|
| Neznámý symbol | 2 | Počet znaků (**ln**) |
| | 2·(**ln**) | Pole znaků symbolu (16bitové Unicode kódování) |
| Chybová konstanta | 1 | Kód chyby |
| Pravdivostní konstanta | 1 | Pravdivostní hodnota |
| Celočíselná konstanta | 4 | Celočíselná hodnota |
| Reálná konstanta | 8 | Reálná hodnota ve formátu IEEE 754 |
| Reálná konstanta | 8 | Číselná hodnota ve formátu IEEE 754 |
| Řetězec znaků | 2 | Počet znaků (**ln**) |
| | 2·(**ln**) | Pole znaků řetězce (16bitové Unicode kódování) |
| Odkaz na buňku | 4 | Index sloupce s počátkem v nule |
| | 4 | Index řádku s počátkem v nule |
| | 1 | Příznaky absolutní adresy sloupce (bit 0) a řádku (bit 1) |
| Uživatelská funkce | 2 | Index do tabulky uživatelských funkcí |

## 5.3 Typy záznamů

### 5.3.1 Záznamy počátku a konce souboru

Záznam typu **BOF**, jehož struktura je popsána v tabulce 48, se nachází vždy na počátku souboru a slouží pro jeho identifikaci. Záznam typu **EOF** se nachází vždy na konci souboru a neobsahuje žádná vnitřní data.

*Tabulka 48:* Struktura vnitřních dat záznamu typu **BOF**

| Velikost (byte) | Obsah |
|---|---|
| 18 | Pole 8bitových znaků: SMARTSHEETDOCUMENT |
| 2 | Verze souboru |

### 5.3.2 Záznam palety barev

Záznam typu **PALETTE** obsahuje záznamy palety barev jako 56 po sobě následujících čtveřic byte. První byte ve čtveřici je nevyužit a další tři obsahují hodnoty barevných složek RGB. Prvních 8 záznamů palety barev se do souboru neukládá.

### 5.3.3 Záznam s údaji tabulky formátů

Záznam typu **FORMAT**, jehož struktura je popsána v tabulce 49, obsahuje údaje jednoho záznamu tabulky formátů. Index záznamu tabulky formátů s počátkem v nule je určen pořadím příslušného záznamu typu **FORMAT** v souboru mezi všemi záznamy typu **FORMAT**.

*Tabulka 49:* Struktura vnitřních dat záznamu typu **FORMAT**

| Velikost (byte) | Obsah |
|---|---|
| 4 | Nastavení formátování (viz tabulka 1) |
| 2 | Indexy do palety barev pro obsah buňky (viz tabulka 2) |
| 4 | Indexy do palety barev pro okraje (viz tabulka 3) |

### 5.3.4 Záznam s údaji tabulky textových hodnot

Záznam typu **TEXT**, jehož struktura je popsána v tabulce 50, obsahuje údaje jednoho záznamu tabulky textových hodnot. Index záznamu tabulky textových hodnot s počátkem v jedničce je určen pořadím příslušného záznamu typu **TEXT** v souboru mezi všemi záznamy typu **TEXT**. První záznam tabulky textových hodnot se do souboru neukládá.

*Tabulka 50:* Struktura vnitřních dat záznamu typu **TEXT**

| Velikost (byte) | Obsah |
|---|---|
| 2 | Počet znaků (**ln**) |
| 2·**ln** | Pole znaků textového řetězce (16bitové Unicode kódování) |

### 5.3.5 Záznamy se základními hodnotami šířky sloupce a výšky řádku

Záznam typu **DEFAULT_COLUMN_WIDTH** obsahuje hodnotu základní šířky sloupce v obrazových bodech o velikosti 2 byte a záznam typu **DEFAULT_ROW_HEIGHT** obsahuje hodnotu základní výšky řádku v obrazových bodech o velikosti 2 byte.

### 5.3.6 Záznamy s údaji sloupců a řádků

Záznam typu **COLUMN**, jehož struktura je popsána v tabulce 51, obsahuje údaje sloupce na příslušném indexu s počátkem v nule.

*Tabulka 51:* Struktura vnitřních dat záznamu typu **COLUMN**

| Velikost (byte) | Obsah |
|---|---|
| 4 | Index sloupce |
| 2 | Šířka sloupce v obrazových bodech |

Záznam typu **ROW**, jehož struktura je popsána v tabulce 52, obsahuje údaje řádku na příslušném indexu s počátkem v nule.

*Tabulka 52:* Struktura vnitřních dat záznamu typu **ROW**

| Velikost (byte) | Obsah |
|---|---|
| 4 | Index řádku |
| 2 | Výška řádku v obrazových bodech |

### 5.3.7 Záznam s informací o sloučených buňkách

Záznam typu **MERGED_CELLS**, jehož struktura je popsána v tabulce 53, obsahuje indexy sloupců a řádků s počátkem v nule, které určují počáteční a koncové buňky oblasti sloučených buněk.

*Tabulka 53:* Struktura vnitřních dat záznamu typu **MERGED_CELLS**

| Velikost (byte) | Obsah |
|---|---|
| 4 | Index prvního sloupce |
| 4 | Index prvního řádku |
| 4 | Index posledního sloupce |
| 4 | Index posledního řádku |

### 5.3.8 Záznamy s obsahem buněk

Záznam typu **CELL_BLANK**, se strukturou popsanou v tabulce 54, představuje buňku s formátováním, ale bez jakékoli hodnoty.

*Tabulka 54:* Struktura vnitřních dat záznamu typu **CELL_BLANK**

| Velikost (byte) | Obsah |
|---|---|
| 4 | Index sloupce s počátkem v nule |
| 4 | Index řádku s počátkem v nule |
| 4 | Index do tabulky formátů |

Záznam typu **CELL_TEXT**, se strukturou popsanou v tabulce 55, představuje buňku obsahující textovou hodnotu.

*Tabulka 55:* Struktura vnitřních dat záznamu typu **CELL_TEXT**

| Velikost (byte) | Obsah |
|---|---|
| 4 | Index sloupce s počátkem v nule |
| 4 | Index řádku s počátkem v nule |
| 4 | Index do tabulky formátů |
| 4 | Index do tabulky textových hodnot |

Záznam typu **CELL_VALUE**, se strukturou popsanou v tabulce 56, představuje buňku obsahující hodnotu jinou než textovou.

*Tabulka 56:* Struktura vnitřních dat záznamu typu **CELL_VALUE**

| Velikost (byte) | Obsah |
|---|---|
| 4 | Index sloupce s počátkem v nule |
| 4 | Index řádku s počátkem v nule |
| 4 | Index do tabulky formátů |
| proměnná | Pole symbolů hodnoty (viz 5.2) |

Záznam typu **CELL_VALUE**, se strukturou popsanou v tabulce 57, představuje buňku obsahující vzorec a hodnotu jeho výsledku.

*Tabulka 57:* Struktura vnitřních dat záznamu typu **CELL_FORMULA**

| Velikost (byte) | Obsah |
|---|---|
| 4 | Index sloupce s počátkem v nule |
| 4 | Index řádku s počátkem v nule |
| 4 | Index do tabulky formátů |
| proměnná | Pole symbolů vzorce (viz 5.2) |
| proměnná | Pole symbolů hodnoty výsledku vzorce (viz 5.2) |

### 5.3.9 Záznam s definicí uživatelské funkce

Záznam typu **USER_FUNCTION**, jehož struktura je popsána v tabulce 58, obsahuje definici jedné uživatelské funkce z tabulky uživatelských funkcí. Index záznamu tabulky uživatelských funkcí s počátkem v nule je určen pořadím příslušného záznamu typu **USER_FUNCTION** v souboru mezi všemi záznamy typu **USER_FUNCTION**.

*Tabulka 58:* Struktura vnitřních dat záznamu typu **USER_FUNCTION**

| Velikost (byte) | Obsah |
|---|---|
| 2 | Počet znaků názvu funkce (**ln**) |
| 2·**ln** | Pole znaků názvu funkce (16bitové Unicode kódování) |
| 2 | Počet povinných parametrů |
| 2 | Počet parametrů |
| 2 | Počet proměnných |
| proměnná | Pole proměnných a parametrů (viz tabulka 60) |
| 2 | Počet stavebních prvků |
| proměnná | Pole stavebních prvků (viz tabulka 59) |

Stavební prvky uživatelské funkce (viz 4.2.2) jsou v rámci záznamu typu **USER_FUNCTION** zapsány v poli, jehož prvky mají podobu uvedenou v tabulce 59, přičemž první čtyři údaje jsou vlastní všem typům stavebních prvků, zbylé se liší podle typu stavebního prvku. Pořadí prvků tohoto pole odpovídá pořadí stavebních prvků dané uživatelské funkce.

*Tabulka 59:* Struktura prvků pole stavebních prvků

| | **Velikost (byte)** | **Obsah** |
|---|---|---|
| **Stavební prvek** | 2 | Identifikátor stavebního prvku |
| | 2 | Index stavebního prvku |
| | 2 | Počáteční index stavebního prvku |
| | 2 | Koncový index stavebního prvku |
| Vrácení hodnoty | 2 | Index zdrojového parametru nebo proměnné |
| Přiřazení konstanty | 2 | Index cílového parametru nebo proměnné |
| | proměnná | Pole symbolů hodnoty konstanty (viz 5.2) |
| Přiřazení hodnoty | 2 | Index cílového parametru nebo proměnné |
| | 2 | Index zdrojového parametru nebo proměnné |
| Unární operace | 2 | Index cílového parametru nebo proměnné |
| | 2 | Identifikátor operace |
| | 2 | Index parametru nebo proměnné operandu |
| Binární operace | 2 | Index cílového parametru nebo proměnné |
| | 2 | Identifikátor operace |
| | 2 | Index parametru nebo proměnné levého operandu |
| | 2 | Index parametru nebo proměnné pravého operandu |
| Funkce | 2 | Index cílového parametru nebo proměnné |
| | 2 | Identifikátor funkce |
| | 1 | Příznak, zda se jedná o uživatelskou funkci či nikoliv |
| Konstanta | 2 | Pole symbolů hodnoty konstanty (viz 5.2) |
| Hodnota | 2 | Index zdrojového parametru nebo proměnné |
| IF THEN | 2 | Index parametru nebo proměnné podmínky |
| IF THEN ELSE | 2 | Index stavebního prvku ELSE |
| | 2 | Index parametru nebo proměnné podmínky |
| WHILE DO | 2 | Index parametru nebo proměnné podmínky |

Parametry a proměnné uživatelské funkce (viz 4.2.1) jsou v rámci záznamu typu **USER_FUNCTION** zapsány v poli, jehož prvky mají podobu uvedenou v tabulce 60. Pořadí prvků tohoto pole odpovídá pořadí záznamů v tabulce parametrů a proměnných dané uživatelské funkce.

*Tabulka 60:* Struktura prvků pole parametrů a proměnných

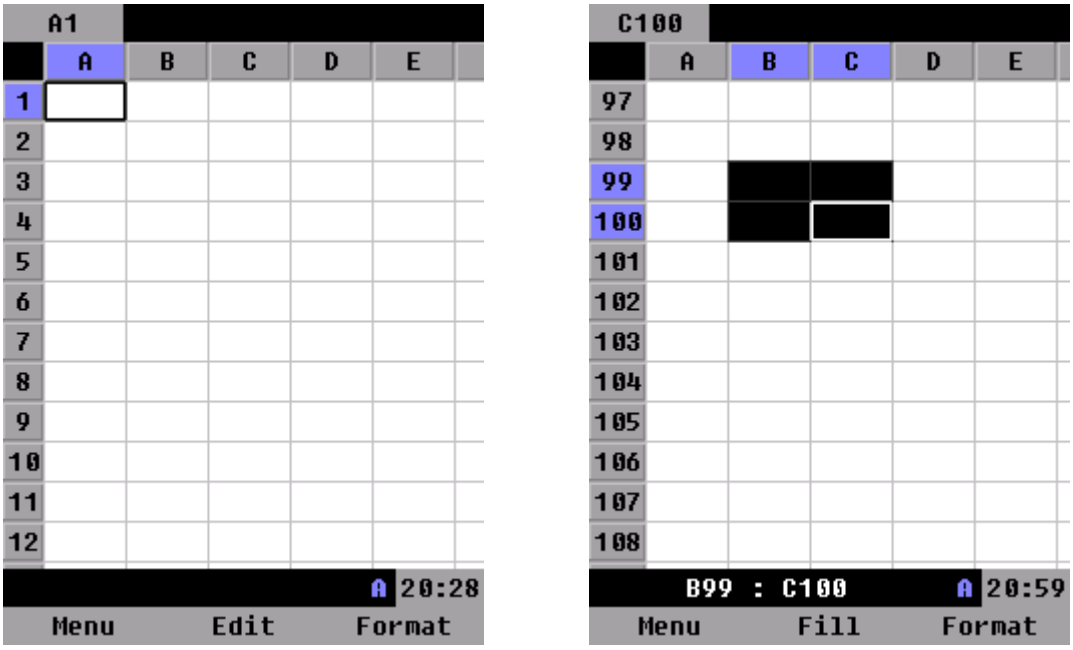
| **Velikost (byte)** | **Obsah** |
|---|---|
| 2 | Počet znaků názvu parametru nebo proměnné (**ln**) |
| 2·**ln** | Pole znaků názvu parametru nebo proměnné (16bitové Unicode kódování) |
| proměnná | Pole symbolů základní hodnoty parametru (viz 5.2) |

# 6 Změny v uživatelském rozhraní

V této kapitole jsou popsány změny v uživatelském rozhraní oproti předchozí verzi aplikace SmartSheet. Součástí je také popis nových komponent pro vstupní formuláře.

## 6.1 Kontextové klávesy

Předchozí verze aplikace SmartSheet nepodporovala kontextové klávesy, které by měnily svou funkci podle stavu, ve kterém se aplikace právě nachází. Spodní řádek s popisem aktuální funkce kontextových kláves, jak je zobrazen v příkladu na obrázku 29, zcela chyběl.




*Obrázek 29:* Popis kontextových kláves při výběru jedné buňky (vlevo) a více buněk (vpravo)

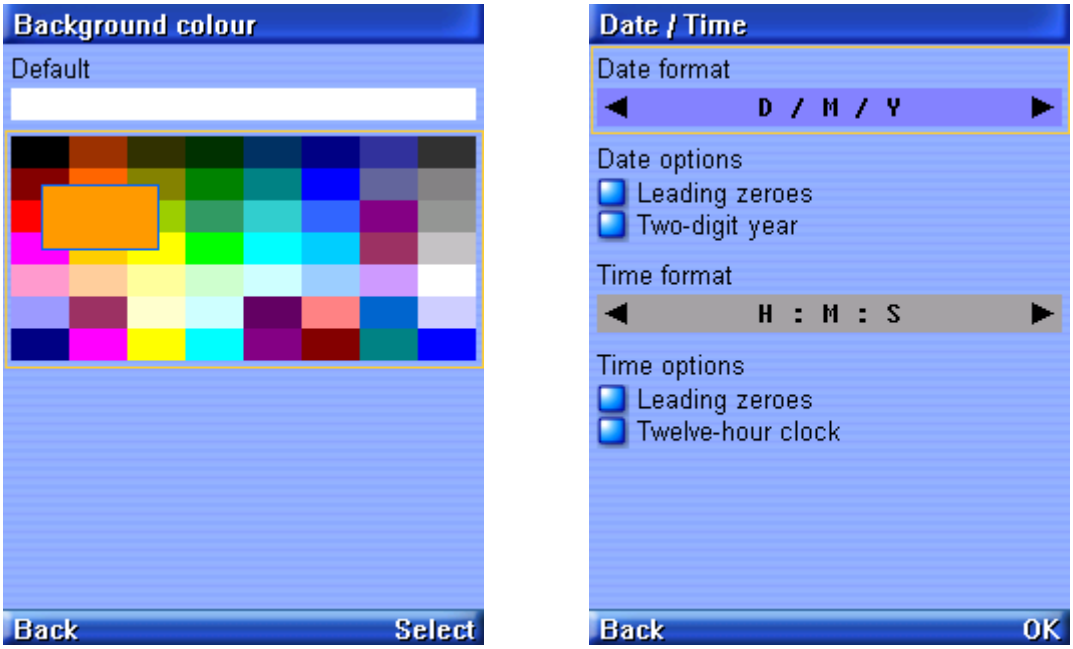
Problémem může být skutečnost, že programové kódy kontextových kláves nejsou součástí J2ME specifikace. Pokud by mobilní zařízení používalo jiné programové kódy než je běžné, kontextové klávesy nebudou funkční, a proto je vždy funkce přiřazená kontextovým klávesám zajištěna také skrze standardizovanou telefonní klávesnici.

Ačkoliv přidání řádku s popisem funkcí kontextových kláves nepatrně zmenšilo výšku plochy pro zobrazení samotného obsahu tabulky, další vlastnost aplikace naopak zvětšila šířku této oblasti pro část tabulky, jelikož sloupec s čísly řádků v této verzi samostatně přizpůsobuje svou šířku podle nejširšího, právě zobrazeného, čísla řádku (viz obrázek 29).

## 6.2 Nové komponenty pro vstupní formuláře

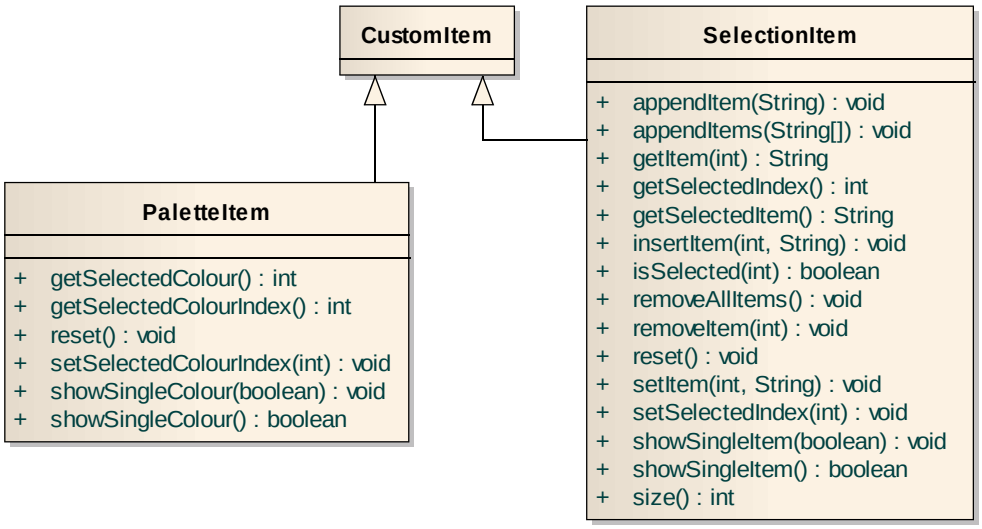
Pro pohodlnější ovládání vstupních formulářů byly vytvořeny dvě nové komponenty. První komponenta slouží pro zobrazení palety barev (viz 2.3) a výběr barvy. Je reprezentována třídou **PaletteItem**. Umožňuje dva způsoby zobrazení, a to zobrazení celé palety barev nebo pouze jedné programově zvolené barvy. Při zobrazení celé palety barev může uživatel kurzorovými klávesami vybrat požadovanou barvu.

Druhá komponenta umožňuje zobrazení a výběr ze seznamu textových řetězců a je reprezentována třídou **SelectionItem**. Jsou možné dva způsoby zobrazení. První zobrazuje všechny položky naráz, což je praktické pouze v případě malého počtu položek, a druhý zobrazuje pouze jedinou, právě zvolenou položku, s možností zobrazení dalších položek pomocí kurzorových kláves. Příklad použití obou komponent je na obrázku 30.




*Obrázek 30:* Příklad použití komponenty **PaletteItem** (vlevo) a **SelectionItem** (vpravo)

Obě třídy dědí z abstraktní třídy **CustomItem** (viz obrázek 31), která je pro účely tvorby nových komponent určena. Třída **PaletteItem** vlastní metody pro nastavení a zjištění aktuálně zvolené barvy. Třída **SelectionItem** obsahuje metody pro úpravy seznamu řetězců a získávání obsahu jednotlivých položek.




*Obrázek 31:* Diagram zobrazující hierarchii nových komponent

# 7 Závěr

Výsledkem této diplomové práce je nová verze aplikace SmartSheet, která zlepšuje některé již existující vlastnosti aplikace a také přidává novou funkcionalitu.

Změny v datových strukturách a ve správě dat mají za následek efektivnější využití paměti. Vytvoření hierarchie tříd pro buňky s různým druhem obsahu a zavedení sdíleného formátování buněk a sdílených textových řetězců umožňuje využít větší část paměťového prostoru pro samotný obsah tabulky. Jelikož je cílovou platformou aplikace mobilní platforma J2ME, kde velikost dostupné paměťi bývá velmi omezena, jedná se o důležitou změnu oproti předchozí verzi.

Přidáním podpory importu a exportu souborů XLS aplikace Microsoft Excel se výrazně rozšířila praktická použitelnost aplikace. Nyní je možné importovat nejen obsah buněk, ale také jejich formátování. Po provedených změnách je možné hodnoty a formát buněk opět zapsat do souboru XLS. Aplikace tedy již není odkázána na přenos dat pouze pomocí souborů CSV.

Možnost vytvářet uživatelské funkce tvoří doplněk ke standardnímu zadávání vzorců do buněk. Skrze uživatelské funkce je možné při výpočtech využít řídících struktur, jakými jsou podmínky a cykly. Do funkcí lze zapouzdřit často používané části vzorců nebo celé vzorce. Tvorba uživatelských funkcí má zcela vizuální charakter, což je při omezených podmínkách vstupu na mobilním zařízení výhodou.

Vlastní formát souborů byl zcela přepracován, tak aby byl pružnější a snáze se udržovala jeho kompatibilita. Zvolený binární způsob uložení dat představuje výhodný kompromis mezi rozšiřitelností a rychlostí zpracování.

Uživatelské rozhraní bylo mírně přepracováno a to především v oblasti struktury nabídek. Byla přidána podpora kontextových kláves a také byly vytvořeny nové komponenty pro vstupní formuláře, které je činí přehlednějšími a vstup údajů pohodlnějším.

Vývoj budoucích verzí aplikace by se soustředil především na rozšiřování knihovny vestavěných funkcí pro použití ve vzorcích a na úpravy uživatelského rozhraní.

# Seznam použité literatury

[1] KUTÁČ, Pavel. J2ME tabulkový procesor. 2008. 31 s. Bakalářská práce. VŠB-TUO.

[2] Windows Compound Binary File Format Specification [online]. Microsoft Corporation, 2007. Dostupné z WWW: <http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD-4342ED7AD886/WindowsCompoundBinaryFileFormatSpecification.pdf>.

[3] Microsoft Office Excel 97-2007 Binary File Format (.xls) Specification [online]. Microsoft Corporation, 2007. Dostupné z WWW: <http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD-4342ED7AD886/Excel97-2007BinaryFileFormat(xls)Specification.pdf>.

[4] RENTZ, Daniel. Microsoft Compound Document File Format [online]. OpenOffice.org, 2007. Dostupné z WWW: <http://sc.openoffice.org/compdocfileformat.pdf>.

[5] RENTZ, Daniel. Microsoft Excel File Format : Excel Versions 2, 3, 4, 5, 95, 97, 2000, XP, 2003 [online]. OpenOffice.org, 2008. Dostupné z WWW: <http://sc.openoffice.org/excelfileformat.pdf>.

# Seznam příloh

**Příloha A** – kompaktní disk obsahující následující položky:

- text diplomové práce ve formátech OpenDocument Text a PDF
- uživatelská příručka aplikace SmartSheet ve formátu PDF
- zdrojové kódy a spustitelná verze aplikace SmartSheet